DHQ: Digital Humanities Quarterly
2020
Volume 14 Number 3

Reassessing the locus of normalization in machine-assisted collation

David J. Birnbaum <djbpitt_at_gmail_dot_com>, University of Pittsburgh
Elena Spadini <elena_dot_spadini_at_unil_dot_ch>, Université de Lausanne

Abstract

In this essay we explore the process of textual normalization in the context of machine-assisted collation, which is a common operation in digital textual scholarship. The Gothenburg modular architecture for computer-assisted collation situates normalization as the second stage within a five-stage computational pipeline, where it contributes to improving the eventual alignment. In this essay we argue that normalization, in fact, contributes not only to the alignment, but also to the interpretation of the texts. Furthermore, it occurs at several moments between transcription and rendering, and should not be regarded as happening all at once and only in one location.

Introduction

In this essay we explore the process of textual normalization in the context of machine-assisted collation, which is a common operation in digital textual scholarship. Collation, that is, the alignment of versions of (traditionally called "witnesses to") the same work, is used by scholars for studying the textual transmission or the genetic process of a work, often as a step in the preparation of a scholarly edition of that work. Machine-assisted collation refers to the use of computers for performing, in full or in part, this alignment. As such, it has a long history, as long as that of the Digital Humanities [Nury and Spadini 2020]: over the past sixty years, not only have the machines at our disposal changed, but so have scholars' understanding of the collation process, and the most refined computational model of collation available today is the one devised in 2009 in Gothenburg by the developers of the collation software Juxta and CollateX and by the textual scholars of the COST Action Open Scholarly Communities on the Web and of the program Interedition [Dekker and Middell 2011] [Dekker et al. 2015] [Bleeker 2017] [Interedition]. The Gothenburg modular architecture for computer-assisted collation is a five-stage computational pipeline, in which **Normalization** constitutes the optional second stage, where it contributes to improving the alignment.[1] 1

In this essay we argue that normalization contributes not only to the alignment, but also to the interpretation of the texts. Furthermore, it occurs at several moments of the collation process, between transcription and rendering, and should not be regarded as happening all at once and only in one location. Our point is not to question the modeling of collation as a computational pipeline, which has led to impressive, productive, and influential results, but to explore the complexity that is elided in the simpler description of **Normalization** in the Gothenburg model, as well as the consequences of that complexity for conceptualizing and implementing a collation project. In fact, for reasons that we explore in detail below, an awareness of how pervasive normalization might be in machine-assisted collation is fundamental to devising a computational workflow that corresponds appropriately to the scholarly requirements of each project. The relationship between our understanding of normalization and our workflow is valid not only for researchers relying on (software that is organized according to) the Gothenburg model, but, more generally, as something to be taken into account by those dealing with machine-assisted collation. That is, the Gothenburg model offers a foundation for discussion, but our argument about the importance and ubiquity of normalization is not dependent on it. Furthermore, as often happens with the computational counterpart of a scholarly practice, machine-assisted collation tends to make explicit the assumptions implicit in manual collation: the study proposed here is largely applicable to collation in general, whether some operations are performed mentally or computationally. For this reason, the essay is addressed not only to textual 2 scholars, but also to developers, who can find here a thorough analysis of the complex procedures implied in machine-assisted collation, with a particular focus on the different kinds of normalization that happen throughout the process.

The essay first lays out preliminary remarks about what normalization is, its manifold forms and purposes, and provides an overview of the Gothenburg model. That model supplies the architecture for what follows: for each stage of the model, the potential role of normalization is discussed and illustrated with examples. In the conclusions, we summarize how this study contributes to a reconsideration of both the role and significance of normalization in collation and digital textual scholarship and, more narrowly, of the Gothenburg model of machine-assisted collation. 3

Normalization

In the context of machine-assisted collation as formalized by the Gothenburg model, described below, **Normalization** refers to the creation and use of shadow copies of word tokens when performing alignment. A common and simple type of normalization is case-folding. For example, although “cat” and “Cat” are not string-equal, collation processing might create a lower-cased shadow copy of each word in the text before performing the comparisons needed for alignment. As a result, although the output might retain the original casing, the alignment engine would use the shadow copies for comparison, and would therefore recognize that the difference between “cat” and “Cat” is not significant for alignment purposes. 4

In this essay we use **Normalization** (capitalized and embolded) to refer to the second, optional, stage of the Gothenburg model, in which the differences in witness tokens that should be ignored for alignment purposes are neutralized, or “normalized,” prior to alignment. We use normalization (without special capitalization or other formatting) to refer to normalization as a general concept, present in many forms distributed over several stages of the pipeline, and affecting not only Stage 2 (**Normalization**), but also, in particular, Stage 0 (**Transcription**), Stage 1 (**Tokenization**), and Stage 4 (**Analysis**). The distinction is instrumental for the main argument of this essay, anticipated above, which is that normalization does not occur only at the **Normalization** stage, and it is, instead, pervasive in machine-assisted collation. 5

Normalization can be performed at different textual levels and it affects many types of textual variation. For example, in addition to the case-folding described above, researchers might wish to neutralize the **graphemic** distinction between Latin alphabet regular “s” (U+0073 LATIN SMALL LETTER S) and long “ſ” (U+017F LATIN SMALL LETTER LONG S). **Orthographic** variation may also transcend the graphemic level, as in the orthographic distinction between US English “color” and British “colour,” or the logographic replacement, rooted in homonymy, of the English preposition “to” by the digit “2” in a telegraphic communicative style popular on social networks. **Morphological** variation includes differences in categories like tense, gender, number, and others; for example, a collation operation might want to recognize that different inflected forms of the same lexeme (such as English “is” and “are,” which have no characters in common) may be textual variants, and are therefore potential candidates for alignment. **Lexical** variants might be identified by semantic features; for example, if one manuscript witness reads “books and magazines” and another reads “journals,” we might want to align “journals” with “magazines,” rather than with “books,” because “journals” is semantically closer to “magazines” than it is to “books.” For multilingual alignment of witnesses in languages with greatly different orthographies (such as Greek [Greek script] and Old Church Slavonic [Glagolitic or Cyrillic script]), normalization might take the form of recording only the part of speech in the shadow copy, since in this research context, part of speech is a better predictor of the alignment of variants than any modification of the orthographic strings [Birnbaum and Eckhoff 2018] 6

Normalization, in a nutshell, makes it possible to identify phenomena on multiple orthographic and linguistic levels and use them to create surrogates for the literal word tokens that then shape and interpret the results of collation, tacitly neutralizing and ignoring other distinctions that are present in the literal tokens. During the process of normalization, an original form is replaced (or, rather, shadowed, since the original form is typically retained and is available for use in the eventual rendered output) by a normalized form. The point of this deliberate neutralization is that different original forms that are normalized (or, in computational terms, that *hash*) to the same value are deemed to be the same at a certain point of the process of machine-assisted collation. 7

Machine-assisted collation: the Gothenburg model

Within a computational pipeline, the output of one process becomes the input to the next, much as water may flow through plumbing that is constructed by concatenating small pieces of piping [McIlroy 1964].[2] Under the Gothenburg model, the digitized text of each witness is piped first through **Tokenization** (Stage 1) and then through **Normalization** (Stage 2), at which point it is joined with all other witnesses to constitute the complex input into **Alignment** (Stage 3). The single output of **Alignment** is then piped through **Analysis** (Stage 4) and **Visualization** (Stage 5) to generate the eventual output of the collation operation. This is illustrated in Figure 1, below, where three witnesses are collated, and Stage 2 (**Normalization**) is highlighted in green.[3] 8

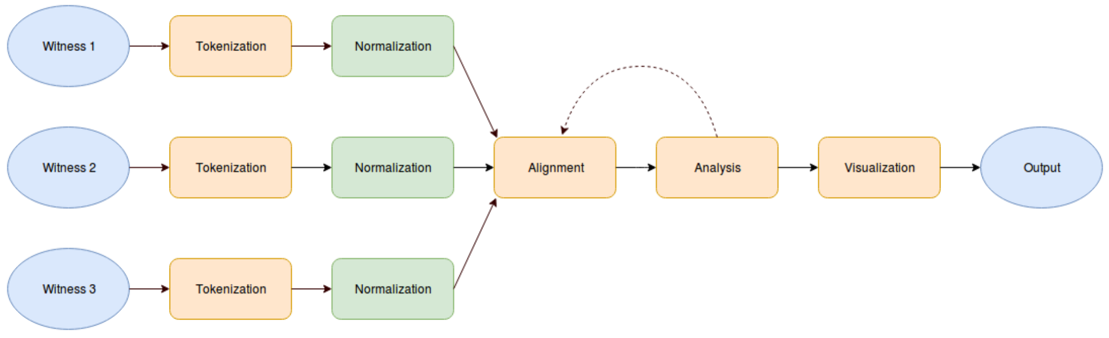

Figure 1. Gothenburg model.

The key contribution of the Gothenburg model to our understanding of collation is its modularization of the process in a way that facilitates customization[4] and is agreed upon by a community of users.[5] Instead of a monolithic black box that ingests witness input and extrudes a collation as a single process (that is, a degenerate pipeline with a single step), a collation system built according to the Gothenburg model, such as CollateX, can expose hooks that support developer intervention at any stage of the process without a need to interact explicitly with the other stages. For example, a user can replace the default CollateX **Normalization** with a customized alternative (see below) without having to touch (or, for the most part, even know about) the implementation of the other Gothenburg stages. As long as the input and output format of each stage is documented, replacing a step in the pipeline is an autonomous operation, that is, one that does not require accommodation elsewhere in the system. In this report we rely for illustrative examples on CollateX, which may be considered the reference implementation of the Gothenburg model.[6] 9

Discussion

This essay challenges the characterization of normalization in the Gothenburg model as something that happens in a single location in the chain of processes (at Stage 2, called **Normalization**). Instead, as we illustrate below, normalization may be performed in four different stages within the overall collation process, not only to improve the alignment, but also to annotate the sources with a layer of information that may be vital for subsequent analysis. In the following description of the five stages of the Gothenburg model (preceded by **Transcription**, which we call Stage 0), the notation "(n)" after the section heading means that normalization plays a role in the stage. 10

Stage 0. Transcription for the purpose of collation (n)

Except in the case of born-digital character-based data, witnesses to be input into a collation engine must first be digitized, that is, transcribed, whether manually or with machine assistance (e.g., OCR). In the case of handwritten documents, this transcription entails the conversion of an analog original to a digital character-stream surrogate. This transcription necessarily is not a one-to-one mapping because the handwritten analog original is infinitely variable, while the distinct character types in the digital result are limited by the character inventory. In practice, the level of detail in the transcription (that is, the extent to which it is "diplomatic") depends on the goals and purposes of the edition, and for that reason, "transcripts are best judged on how useful they will be ..., rather than as an attempt to achieve a definitive 11 transcription” [Robinson and Solopova 1993, 19]. Furthermore, Robinson and Solopova continue, because the complete character representation of all features on a handwritten original is both impossible and undesirable, transcriptions “must be seen as fundamentally incomplete and fundamentally interpretative” [Robinson and Solopova 1993, 21].

12 The conflation of multiple physically different written signs into the same single digital characters is an instance of normalization, that is, of determining, sometimes subconsciously, that certain objective differences may (or, perhaps, should) be ignored because they are not relevant informationally.[7] The risk of normalization during manual transcription, though, is that information that is discarded during transcription is not available at later stages of collation, or during subsequent use of the collation output in research. Researchers are nonetheless comfortable with this type of normalization not only because it is inevitable, for the reasons described above, but also because they are confident that the data that they are excluding is not informational. The alternative, that is, trying to transcribe with as little normalization as possible, not only cannot be perfect, but also comes at a cost, because the greater the number of graphic distinctions the researcher tries to preserve during transcription, the greater the price in terms of both efficiency (because the researcher must be alert to more details) and accuracy (because there is more opportunity for distraction, error, and inconsistency) [Robinson and Solopova 1993, 25]. A sensible compromise, especially in situations where photographic facsimiles are available, and therefore reduce the documentary value of a *hyperdiplomatic* transcription, is that the digital transcription should preserve differences in the original orthography that might be needed for adequate rendering (as determined by the goals of the project) or serve as eventual input into computational analysis (which might include alignment within the collation process, subsequent analysis of patterns of variation, or orthographic or linguistic analysis that is not connected explicitly to collation).[8]

13 Normalization during transcription need not be entirely silent. For example, as a way of accommodating both diplomatic and normalized representations during transcription, the Text Encoding Initiative (TEI) makes it possible to couple the two formally where that is sensible and practical, as in the following example from the TEI P5 Guidelines:[9]

```
<l>But this will be a <choice>
   <orig>meere</orig>
   <reg>mere</reg>
</choice> confusion</l>
<l>And hardly shallter we all be <choice>
   <orig>vnderstoode</orig>
   <reg>understood</reg>
</choice>
</l>
```

This strategy makes it possible, outside a collation context, to index or search for words according to their normalized forms while rendering them according to the variant orthography that appears in the source. It is nonetheless the case that even the `<orig>` reading necessarily undergoes some degree of normalization as part of the transcription process.

Stage 1. Tokenization (n)

14 The alignment of segments of text for collation presupposes the division of a single continuous stream of characters into tokens, typically words (however we define them) and punctuation marks, although nothing precludes other forms of tokenization.[10] CollateX, for example, incorporates a default tokenization function that splits the input into sequences of word characters (regex `\w+`) plus any trailing whitespace (regex `\s+`), and further breaks off punctuation marks together with any trailing whitespace into their own tokens (regex `\W+`). In this way, for example, a word followed by a period at the end of a sentence will be recognized as string-equal to the same word without the period in a different witness.

15 How to manage whitespace, which is one of the common issues that must be resolved during tokenization, might also entail forms of normalizations. Although tokenization on whitespace in some other programming contexts (such as the

XPath `tokenize()` function or the Python `str.split()` method with a null *separator* value) may regard whitespace as a separator between tokens that should be discarded, default tokenization in CollateX, for example, keeps the whitespace as the trailing part of the preceding token, so that nothing is discarded during tokenization.[11] This whitespace is subsequently removed in CollateX by the default **Normalization** stage unless that is overridden (see the following section), thus preserving the strict Gothenburg model distinction between **Tokenization** and **Normalization**. Yet regarding all whitespace characters as equivalent for tokenizing the input, and regarding sequences of multiple whitespace characters as equivalent to a single space, both of which are part of the default implementation of **Tokenization** in CollateX, are forms of normalization, that is, situations where forms that are not string-equal are nonetheless deemed to be equivalent for a particular purpose.

Thanks to the modular architecture of the Gothenburg model adopted by CollateX, which provides hooks into the **Tokenization** stage, users can replace the default **Tokenization** with custom code without needing to know about or otherwise touch the code that manages the other stages of processing.[12] For example, where punctuation is not intended to be used in alignment but must nonetheless remain available in the final output, it is possible to tokenize only on sequences of whitespace characters, regarding punctuation not as a separate token, but as part of the word token that precedes or follows it, where it can then be ignored (during Stage 2) for the alignment (during Stage 3). 16

Stage 2. Normalization (n)

The Gothenburg model regards **Normalization** primarily as a way of improving Alignment by recognizing that tokens that are not string-equal should nonetheless be deemed to be equivalent for the purpose of **Alignment**: 17

> It might suffice to normalize the tokens’ textual content such that an exact matching of the normalized content yields the desired equivalence relation. For instance, in many cases all tokens of the text versions are normalized to their lower-case equivalent before being compared, thereby making their comparison case insensitive. Other examples would be the removal of punctuation, the rule-based normalization of orthographic differences or the stemming of words. [CollateX Doc]

As implemented within CollateX, a token is a complex object that contains at least two properties: a t (“text”) property, which represents the string value of the token after **Tokenization**, and an n (“normalized”) property, which represents the result of applying **Normalization** to the t value. The software uses agreement among the n properties to recognize tokens that should be aligned. The t and n values are created either by the built-in default **Tokenization** and default **Normalization** operations,[13] or by custom operations implemented by the researcher to replace the defaults. The fact that each token has the t and n properties means that CollateX exposes the output of **Tokenization** alongside the output of **Normalization**, providing hooks for customization that make the software useful for scholars with a wide variety of editorial requirements.

Normalization at Stage 2 in the Gothenburg process serves a specific purpose: it is performed so that variation that the editor considers unimportant for alignment will not influence Stage 3, when **Alignment** is performed.[14] This is different from the purpose of normalization during **Transcription** (which we called Stage 0, above) or during **Analysis** (Stage 4, below), and it has different consequences. If the editor normalizes the text during **Transcription** and does not record the non-normalized forms, those forms become irretrievable. **Normalization** at Stage 2 of the Gothenburg model, however, is non-destructive, since the normalized form is created as a shadow copy of the original, and not as a replacement for it. As Bleeker explains: 18

> “normalization” in the context of automated collation is not equivalent to normalization that happens in transcription. For example, editors can transcribe orthographic variation because they consider it important to be preserved in both the transcription and the collation output. However, in the collation process itself, they may want to normalize orthographic variation because they do not want it to influence the alignment. In that case they need to normalize their tokens before inputting them in the collation software. [Bleeker 2017, 94]

Common types of normalization, some of which were mentioned in the introduction, above, are discussed individually in the subsections below. Many of these examples might have been aligned correctly even without normalization because of forced matches, near-matching (see below), or — in situations where the software can decide only arbitrarily among alternatives — by chance.[15] But because of the computational complexity of alignment, normalization that leads to the identification of more exact matches (of the normalized shadows of the tokens) can nonetheless improve both the accuracy and the efficiency of the overall performance.

Case-folding (orthographic)

In the following example from the *Frankenstein variorum* Project, the case distinction between “SAVILLE” and “Saville” in the third token column is not regarded as significant for **Alignment** purposes:[16] 19

Witness	Tokens			
1818	To	Mrs.	SAVILLE,	England.
1823	To	Mrs.	SAVILLE,	England.
Thomas	To	Mrs.	SAVILLE,	England.
1831	To	Mrs.	Saville,	England.

Table 1. Case distinction in the *Frankenstein variorum* Project

These tokens would have been aligned correctly in any case because they constitute a forced match between “Mrs.” and “England.” But the information stored and processed during the **Normalization** stage is available not only for **Alignment**, but also, for example, for distinguishing, during subsequent **Analysis**, forced matches that are not deemed equivalent from those that are.

Graphemic (orthographic)

In early Cyrillic writing, “з” (U+437 CYRILLIC SMALL LETTER ZE) and “ѕ” (U+0455 CYRILLIC SMALL LETTER DZE) had different pronunciations and distribution in Old Church Slavonic, but came to be used largely interchangeably in early East Slavic writing, including in the *Rus' primary chronicle*.[17] In the last column in the following example from that work (4,8), the Xle witness uses one variant and other witnesses agree on the other, and the editors neutralize the distinction at the **Normalization** stage, so that the forms will be deemed equivalent for purposes of **Alignment**:[18] 20

Witness	Tokens				
Lav	по	семуже	морю	сѣдѧть	варѧ|зи
Tro	по	семуже	морю	сѣдять	варязи
Rad	по |	семоуже	морю	приседѧть	варѧзи
Aka	по	семѹже	морю	приседѧтъ	варѧзи.
Ipa	по	семуже	морю	сѣдѧть	ва|рѧзи
Xle	по	семоуже	морю	сѣдѧт	варѧѕи.
Byč	по	семуже	морю	сѣдять	Варязи
Šax	по	семуже	морю	сѣдять	Варязи
Lix	По	сему&#xao;же	морю	сѣдять	варязи
Ost	По	семуже	морю	сѣдять	Варязи

Table 2. Graphemic variation in the *Rus' primary chronicle*

Spelling (orthographic)

The 1818 and 1823 editions of Mary Shelley's *Frankenstein* regularly use the US-associated spelling of the verbal suffix "-ize," while the 1831 edition regularly uses the British-associated spelling "-ise." In the *Frankenstein variorum* Project, this distinction can be neutralized before performing **Alignment**, as in the case of "tranquillize" and "tranquillise" in the seventh token column in the following example. 21

Witness	Tokens								
1818	for	nothing	contributes	so	much	to	tranquillize	the	mind
1823	for	nothing	contributes	so	much	to	tranquillize	the	mind
1831	for	nothing	contributes	so	much	to	tranquillise	the	mind

Table 3. Spelling variation in the *Frankenstein variorum* Project

The same phenomenon can be observed in the following example, from the *Lancelot* manuscript transmission [Spadini 2016], where Witness A preserves an archaic spelling and Witnesses B and C reflect the modern one: 22

Witness	Tokens			
A	ge	-	te	conois
B	artu	je	te	conois
C	artus	je	te	conois

Table 4. Spelling variation in *Lancelot en prose*

When CollateX fails to find an exact string match during Alignment and there are multiple options for placing a token (in this case, "ge" may be placed in the first or second token column), the software arbitrarily defaults to the leftmost position, which, in the example above, is philologically incorrect.[19] But by normalizing the two different spellings of the pronoun, "ge" in Witness A is aligned correctly: 23

Witness	Tokens			
A	-	ge	te	conois
B	artu	je	te	conois
C	artus	je	te	conois

Table 5. Correct alignment of spelling variants in *Lancelot en prose*

The *Rus' primary chronicle* manuscripts illustrated above vary orthographically for reasons that are both properly orthographic (that is, concerned with scribal conventions for writing correctly) and underlyingly phonetic (insofar as the manuscripts were written at different times, they do not all represent the same state of the language). Identifying all individual neutralizations of letter differences for normalization is not practical because of the complexity of the systems and their relationships to one another. It is, however, possible to overcome this limitation with a normalization scheme that retains only the parts of the tokens that have a high information load with respect to grouping readings into variant sets. For this project we implemented a normalization scheme based on a modification of Soundex [Soundex], recognizing that, in these writing systems, the beginning of the word has a higher information load than the end, consonants have a higher information load than vowels, and some phonetic distinctive features have a higher information load than others. For details and examples see Birnbaum 2015. 24

Digits (orthographic)

The distinction between numbers spelled as digits and those spelled as words may be neutralized for alignment purposes. The following example is from the *Rus' primary chronicle* (1,2), where, in the fourth token column, Ost uses an Arabic numeral; Rad, Aka, and Ipa use a Cyrillic letter to represent a numerical value, followed by a grammatical 25 ending; and the other witnesses spell out the entire number as a word (a different number in Lav than in the others).

Witness	Tokens						
Lav	по	потопѣ.		первие	снве	ноеви.	раздѣлиша
Tro	по	потопѣ		трие	сынове	ноеви	раздѣлиша
Rad	по	потопѣ.		ѓе	сн́ве	ноеви.	разделиша
Aka	по	потопе		.ѓ.е	сн́ве	ноеви.	разделиша
Ipa	по	потопѣ	бо	.ѓ.е.	с́нве	ноеви	роздѣлиша
Xle	Пѡ	потопѣ	оубо	трїе	с́нове	ноеви.	раздѣлиша
Byč	По	потопѣ		трие	сынове	Ноеви	раздѣлиша
Šax	По	потопѣ	убо	трие	сынове	Ноеви	раздѣлиша
Lix	По	потопѣ		трие	сынове	Ноеви	раздѣлиша
Ost	По	потопѣ		3-е	сынове	Ноеви	раздѣлиша

Table 6. Number rendering in the *Rus' primary chronicle*

Punctuation (orthographic)

If what is important to the researcher is to distinguish simply the presence or absence of punctuation, but not the form it takes, different punctuation tokens may be neutralized as a single, generic punctuation token. If punctuation is treated during tokenization as part of the word that precedes it, the punctuation marks may be compared literally, as they appear in the manuscripts, in which case, for example, a word followed by a comma will not match the same word followed by a dot. Alternatively, trailing punctuation can be normalized so as to be ignored during alignment, as in the example below from the *Rus' primary chronicle*. Here in the fourth token column the Rad and Xle readings differ not only in the second letter, but also in the trailing punctuation, which is a dot in Rad and a comma in Xle: 26

Witness	Tokens								
Lav	инд<иꙗ>|	по	ефратъ	рѣку.		вавилонъ.	кордуна.	
Tro	индия	по	ефратъ	реку		вавилонъ	кордуна	
Rad	мидиꙗ.|	и	ефрат	река.	и	вавилон.	кордоуна.	
Aka	мидиа.	и	ефратъ	река.|	и	вавилонъ.	кордᲂуна.	
Ipa	мидиа.	и |	ефратъ	рѣку.	и	вавилонъ.|	кордуна.	
Xle	медїа	и	ефратъ |	рѣка,	и	вавѵлѡн.	кордоуна.	и
Byč	Мидия	по	Ефратъ	рѣку,		Вавилонъ,	Кордуна,	
Šax	Мидия	и	Ефратъ	рѣка	и	Вавилонъ,	Кордуна,	
Lix	Мидия	по	Ефратъ	рѣку,		Вавилонъ,	Кордуна,	
α	Мидия	и	Ефратъ	рѣку	и	Вавилонъ,	Кордуна,	

Table 7. Variation in punctuation in the *Rus' primary chronicle*

Morphological (linguistic)

Morphological variation (e.g., inflectional endings that express number, gender, case, tense, and other linguistic categories) may be neutralized before **Alignment**, similarly to what in corpus linguistics is called *lemmatization* or *stemming*. In the following example the present (Witness A) and future (Witnesses B and C) tense of the verb “chevaucher” (“to ride”) might be normalized in order to obtain an optimal alignment. 27

Witness	Tokens							
A	que	ge	chevauchoie	-	a	tot	mon	poir
B	que	je	chevalcheroie	-	a	tot	mon	pooir
C	si	-	chevalcheroie	la	a	tot	mon	pooir

Table 8. Morphological variation in *Lancelot en prose*

Lexical (linguistic)

In Old French, “pas” and “mie” are negative particles that are used together with the negation adverb “ne”. They have no textual characters in common, but their meaning and syntactic pattern of use is the same, making them candidates for alignment. In the following example, the alignment would be sub-optimal without a normalization of the two forms, because “mie” would be placed into the fourth token column, aligned with “on” in Witness B, instead of in the fifth, aligned with “pas”.[20] If normalization is performed, the alignment is correct, as reflected in the table below: 28

Witness	Tokens				
A	ne	-	doit	-	mie	atorner
B	ne	li	doit	on	pas	atorner
C	ne	il	doit	-	pas	atorner

Table 9. Lexical variation in *Lancelot en prose*

Syntactic role (linguistic)

Consider the following hypothetical alignment example from the CollateX development test suite: 29

Witness	Untokenized text
A	I bought this glass because it matches those plates.
B	I bought those glasses.

Table 10. Example of texts to be aligned

Alignment according to exact orthographic matching would align the two instances of “those,” producing: 30

Witness	Tokens								
A	I	bought	this	glass	because	it	matches	those	plates.
B	I	bought						those	glasses.

Table 11. Problematic alignment of the example in Table 10

Some editors, though, might prefer to align the direct objects, that is, to assign greater weight to the syntactic role than to string matching. This would produce: 31

Witness	Tokens								
A	I	bought	this	glass	because	it	matches	those	plates.
B	I	bought	those	glasses.					

Table 12. Preferable alignment of the example in Table 10

Language (linguistic)

In this section a segment of text from the Old Church Slavonic (OCS) *Codex Suprasliensis* is aligned with a reconstructed Greek parallel text. Because the two languages use different scripts, no type of orthographic normalization would improve string matching, but when a part of speech identifier is used as the shadow normalization property of the tokens, it allows for quite accurate alignment. In the following example, the second gap in the OCS is aligned with a Greek definite article because Greek has articles and OCS does not. The first gap reflects the presence of a personal pronoun in the Greek that happens not to be in the OCS. The tokens that are aligned in the two witnesses belong, pairwise, to the same parts of speech, which is how CollateX knew where to position the gaps [Birnbaum and Eckhoff 2018]. 32

Witness	**Tokens**						
OCS	посълалъ		к	тебѣ	и҆скоусити		ȏусрьдиѥ
Greek	ἀπέσταλκέ	με	πρὸς	σέ	δοκιμάσαι	τὴν	πρόθεσίν

Table 13. Multilingual alignment in the *Codex Suprasliensis* project

Stage 3. Alignment

No normalization is performed during the **Alignment** stage of the Gothenburg model, but the alignment engine has access to the results of the normalization performed in the **Normalization** stage, as well as the less explicit normalization, described above, that happens during **Transcription** and **Tokenization**. And, as is explained below, because the output of **Analysis** (Stage 4) can be cycled back into another iteration of **Alignment**, **Alignment** also may have access to normalization performed during **Analysis**. 33

The input into the **Alignment** stage of CollateX is a set of token sequences, one sequence for each witness, where, as described above, each token has a t property, which represents a transcription of its reading according to the witness, and an n property, which represents a normalized version of the t property.[21] As described above, Normalization is not limited to orthography; the researcher may create, as the n property, any surrogate for the token that will facilitate alignment. The alignment engine then uses only the n property to try to find the best alignment of the tokens, that is, the alignment that corresponds most closely to what a human philologist would produce.[22] 34

Alignment is, to be sure, a more complicated process than simply aligning the tokens with matching n values. On the one hand, in a text of any meaningful length, word tokens are likely to repeat, which means that alignment must associate the correct instances of each value, evaluating and making decisions about many-to-one or many-to-many correspondences. For example, in the following alignment table: 35

Witness	**Tokens**						
A	la	ou	il	conquist	par	sa	chevalerie
B		ou	il	conquist	la	grant	hautesce
C	la	ou	il	conquist	la	grant	hatece

Table 14. Correct alignment of repeated word token “la”

“la” (token columns 1 and 5) occurs once each in witnesses A and B (in different locations) and twice in witness C. The alignment engine matches these instances correctly because, although it operates at a certain level with individual tokens, it also knows about sequences of vertically aligned *blocks.* This means that it has access to the context when deciding which instances of repeated tokens in one witness to align with which instances of matching tokens in the others. 36

Repetition (that is, the occurrence of multiple instances of the same token) is a well-known challenge for collation, and 37

Andrews (2020) proposes an unusual normalization strategy as a way of meeting that challenge. Andrews observes that when a computational process aligns very common words because they are string-equal, they often do not belong to corresponding textual units, which both produces misalignments and increases the computation complexity of the collation process:[23]

> Common readings keep being treated as equivalent, just because they are identical! This often throws off the collation of more substantive words, because the collator doesn't know how to tell what is substantive or not. In a long text with many witnesses, this can throw off a collation [... and ...] the collation algorithm was taking longer and longer. [Andrews 2020]

To avoid the spurious alignment of tokens that, although string-equal, are so common that their correspondence is more likely to be accidental than philologically meaningful, Andrews replaces these common tokens, as well as punctuation tokens, with random strings during **Normalization**. As a consequence of this random replacement, the normalized surrogates do not agree with anything during **Alignment**, and therefore do not complicate or confuse that process. This enables substantive agreements to govern the alignment, improving both the quality of the output and the speed of the operation (Andrews reports that it "made the process run 4–5 times faster").

On the other hand, the alignment engine may also choose to align non-matching tokens that fall in the same position, as in the fourth position in the example below. 38

Witness	Tokens							
A	lors	abat	la	destre	manicle	de	son	hauberc
B	lors	abat	la	senestre	manicle	de	son	hauberc
C	lors	abat	la	senestre	manicle	de	son	hauberc

Table 15. Forced match in the alignment

The token alignment in seven of the eight columns reflects exact string equality across all witnesses, but "destre" (right) and "senestre" (left), in the fourth token column, are not an exact match. This alignment is forced by the context: given that the "la" tokens before and the "manicle" tokens afterwards are aligned as exact matches in all witnesses, the tokens between them fall into alignment by default, a situation we call a *forced match*.[24] **Alignment** is further complicated by token transposition, where tokens in one order in one witness may need to be aligned with tokens in a different order in another witness, while nonetheless retaining each witness's token order. 39

Finally, the algorithms most commonly used to implement alignment routines, such as Needleman-Wunsch [Needleman and Wunsch 1970] or the Dekker [Dekker and Middell 2011] algorithm in CollateX, reduce the computational complexity of the alignment process only at a cost to the philological integrity of the output: 40

> These two algorithms follow the principle of "progressive alignment," which means that they do not compare all witnesses at the same time. Instead, they first compare two witnesses, store the result of that comparison in a so-called variant graph and then progressively compare another witness against that graph, at every turn merging the result of that comparison into the graph until all witnesses are merged. This progressive alignment method reflects the idea of "dynamic programming," which takes a complicated problem and breaks it down in smaller sub-problems. In this case, the complicated task of multi-witness alignment is broken down into a repetition of the relatively easier task of two-witness alignment. A downside of progressive alignment is that, apparently, the order in which the witnesses are compared influences the final result. That is, the final alignment of three witnesses A, B, and C may differ if Witness C is compared against the variant graph of Witness A and B, or if Witness B is compared against the variant graph of Witness A and C. [Bleeker 2017, 95]

The preceding means that progressive alignment cannot be considered an optimal strategy for multiple witness collation, since the philologically optimal alignment of the witnesses obviously cannot logically depend on the order in which the philologist (or algorithm) looks at them [Spadini 2017, 348].[25] At the same time, the computational complexity of comparing all witnesses to all other witnesses simultaneously means that optimal order-independent multiple-witness alignment is currently an unsolved problem in computer science,[26] and progressive alignment algorithms, despite their limitations, represent the current state of the art.

Alignment algorithms typically test only for exact matches between tokens, because looking for the closest but inexact match is prohibitively expensive computationally. For that reason, the CollateX implementation of near matching, which we describe below, is performed in the **Analysis** stage and then recycled into another instance of **Alignment**. What is important in this discussion of Stage 3 is that, to the extent that near matching can be said to entail a type of normalization, it happens not in Stage 2, but, rather, in Stage 4, after an initial alignment has already been constructed. 41

Stage 4. Analysis and interpretation (n)

The 2009 meeting that led to the elaboration of the Gothenburg method did not produce a final report or white paper, and in the absence of an authoritative definition, the type of analysis that occurs at Stage 4 has been interpreted variously as both computational postprocessing and manual, human intervention between the output of **Alignment** (Stage 3) and the input into **Visualization** (Stage 5). In situations where the alignment is sub-optimal and cannot be repaired algorithmically, human intervention becomes necessary. The cost of this intervention with respect to the workflow, though, is that it introduces changes into a generated interim artifact, rather than into the *base view*, that is, the transcribed witness files that serve as the initial input into the processing pipeline. Any changes introduced manually beyond the base view mean that if the collation operation must be re-run from the beginning (for example, if a new witness is added, or if editorial decisions about word division change), the manual effort is lost, and must be repeated. In other words, manual intervention creates a new, derived base view, one that can no longer be generated from the original base view entirely by the computational pipeline. 42

The purpose of the **Analysis** in Stage 4 in the Gothenburg model, whether automated or implemented manually, is described clearly in the CollateX documentation. In the case of sub-optimal output from the **Alignment** stage: 43

> [a]n additional [...] analysis of the alignment result [...] may alleviate that deficiency by introducing the possibility of a feedback cycle, in which users edit the alignment and feed their knowledge back into the alignment process for another run delivering enhanced results.[27]

Postprocessing performed at Stage 4, in addition to possibly improving the eventual alignment, may also be used to infer knowledge from the information added to the original texts through the first three Stages of the collation pipeline. That is, the end products of a full collation pipeline, from **Tokenization** through **Visualization**, may include not only a critical edition with variation, but also, for example, summary analytic reports about the tradition, whether textual (e.g., in statistical tables) or graphic (e.g., in charts or diagrams).

In this section we identify two types of computational analysis, both involving normalization, that are located at Stage 4 of the Gothenburg model. These are 1) near matching, 2) the analysis of patterns of agreement within alignment sets. Near matching is intended to improve the alignment, which is to say that the output of this Stage 4 process, after modification for near matching, is fed back into Stage 3 for realignment. The analysis of patterns of agreement is intended to enrich and customize the collation output, and is fed into Stage 5 (**Visualization**). Especially in light of the absence of clear guidance in the literature about the Gothenburg model concerning the **Analysis** stage, the two types of computational analysis discussed here should be regarded as only some of the possibilities available at this stage of the collation pipeline. 44

Near matching

The term “near matching”, sometimes called “approximate” or “fuzzy” matching, refers to the identification of the closest match for alignment purposes, specifically in situations where there is no exact match.[28] As was noted above, when CollateX compares the n properties of tokens to create an initial alignment, it looks only for exact string matches. That 45 is, to align a token it asks not “what is the closest match?,” but, rather, “is there an exact match?” The computational complexity of checking for the closest match is sufficiently greater than the complexity of checking for an exact match that it would not be realistic to perform an entire alignment operation by computing all closest matches in all alignment positions. Furthermore, as noted earlier in the context of forced matches, as long as the alignment process finds a sufficient number of exact matches, a large number of inexact matches are likely to be forced into proper alignment anyway.

A situation that is susceptible to incorrect alignment involves the following two features: 46

1. One witness has fewer tokens than another, which means that there will be a gap in the alignment, where a token in the longer witness does not have a corresponding token in the shorter one.
2. There is a token in the shorter witness that is adjacent to the gap and that does not have an exact match in any of the alignment positions where it could be placed.

When both of the preceding conditions are met, an alignment engine that relies on exact matching is unable to decide where to position the gap, that is, whether to push a token with two or more possible inexact adjacent alignment points to the left or to the right. It is at this stage that near matching can be used to resolve the uncertainties, and because the number of comparisons for this type of limited, strategically targeted near matching is exponentially less demanding computationally than what would be required to perform near matching on all tokens during the initial alignment, it does not impose an unacceptable delay. 47

As an example, if we need to align “The gray koala” with “The grey koala” (note the different spellings of the color adjective) and we have not normalized one of the spellings to the other, the color words will nonetheless wind up aligned correctly because they represent a forced match between the perfect matches of “The” to the left and “koala” to the right. But suppose we have to align “The gray koala” with “The fluffy grey koala,” that is, suppose we have to decide whether to align “gray” in the first witness with “fluffy” or with “grey” in the second. Without near matching, CollateX has to guess, and its arbitrary strategy is to push the token in question to the left, which will produce an incorrect result: 48

Witness	Tokens			
A	The	gray		koala
B	The	fluffy	grey	koala

Table 16. Problematic alignment of similar word tokens

The alignment a scholar would prefer is: 49

Witness	Tokens			
A	The		gray	koala
B	The	fluffy	grey	koala

Table 17. Correct alignment of similar word tokens

With near matching, however, the **Analysis** stage could determine that “gray” is more similar, in terms of its string value, to “grey” than it is to “fluffy,” and recycle this information into a second, targeted, **Alignment** process that would position the token accordingly. This is how near matching works in CollateX.[29] 50

The following example of the use of near matching in the **Analysis** stage to correct misalignments in the **Alignment** stage is from the *Rus' primary chronicle* (3,5). In the table below, which presents the output of CollateX without near matching, the last token of Tro is misaligned. It is not string-equal to any token in either of the last two columns (note the fourth letter of the word, which does not match the fourth letter of the words in the last column of the other witnesses), so CollateX arbitrarily pushes it to the left, even though it is a closer match with the tokens to the right. 51

Witness	Tokens				
Lav	гаръмати	тавріани.	сируфьꙗ.		фраци.
Tro	гаръмати	тавриани	скуфиа	фраки	
Rad	сармати	таврилни	скоуфиа	и	фраци
Aka	сармати.	тавріани	скоуфіа.	и	фраци
Ipa	сармати.	тавриани.	скуфиꙗ.		фраци.
Xle	сармати.	тавріани.	скѵфіа		фраци.
Bych	Саръмати,	Тавриани,	Скуфиа,		Фраци,
Shakh	Сармати,	Тавриани,	Скуфия,		Фраци,
Likh	Саръмати,	Тавриани,	Скуфиа,		Фраци,
Ost	Сармати,	Тавриани,	Скуфия,		Фраци,

Table 18. Incorrect alignment of similar word tokens in the *Rus' primary chronicle*

Near matching is optional in CollateX, and without it we get the output above. If we turn on near matching, though, the last token in Tro is moved to the right because it is a closer match to tokens in the right column than to any in the left. 52

Near matching in CollateX uses the python-Levenshtein package to calculate similarity, which means that it does not have access to variation information beyond the character level, such as lexical substitution. A small Levenshtein distance can identify situations where scribes might have misread or miswritten individual letters and introduced a small, local corruption. But scribes might also intervene consciously to replace one entire word with another, and in such situations, the Levenshtein distance between the words is not necessarily a useful measure of *editorial* distance, that therefore also not necessarily a useful predictor of the optimal alignment. In such cases, a comparison metric that did not rely solely on Levenshtein character edit distance, and that also had access to other properties, might achieve a philologically preferable alignment. 53

Patterns of agreement

Normalization is ultimately a decision about when objectively different written forms should be deemed equivalent *for a specific purpose*. At the **Alignment** stage (whether initially or with recycled input after near matching, as described above), that specific purpose is to determine which tokens should be considered part of the same variant set, whatever the category of variation. For subsequent philological research, however, it is common to distinguish between *formal* and *substantive* (sometimes called *insignificant* and *significant*) variants, where only the latter are traditionally regarded as useful for stemmatic purposes: 54

> philological judgement is deployed to distinguish “significant” from “insignificant” textual variation — that is, to select those variants that are more or less likely to betray information about the exemplar from which a given text was copied.[30] [Andrews 2016, 523]

As described above, **Normalization** for **Alignment** purposes (Stage 2, feeding into Stage 3) in CollateX, by default, writes a shadow value to be used during alignment into an n property on the token. A user-designed replacement for the default normalization routine built into Stage 2 could customize the modifications employed to create the n property, and could also add other properties to the tokens — for example, in addition to orthographic normalization in the n property, it might identify the lemma, the part of speech, and the morphology, and write those into l, p, and m properties. The built-in **Alignment** stage of CollateX ignores these other properties, but it passes them along the pipeline, which means that they are accessible at later stages. Recalling that normalization is “a decision about when objectively different written forms should be deemed equivalent *for a specific purpose*,” we might perform a different type of normalization (indeed, a form of interpretation) during **Analysis** in Stage 4, comparing tokens that have already been aligned with one another to determine whether their l, p, and m properties (or some subset of those) agree. This type of analysis would enable us to distinguish, within a variant set, which readings agree completely (all properties), which agree in 55 traditionally significant properties but differ in insignificant ones (e.g., agree in l, p, and m, but not n), and which differ in significant properties (e.g., disagree in l, p, or m).[31]

In the following example, the tokens *ge* and *je* both have the same (hypothetical) substantive properties: l (lemma) = “je,” p (part of speech) = “personal pronoun,” and m (morphology) = “first person singular.” But because the original reading (t property) is different, we deduce that the variant is orthographic or phonological, which we consider formal, rather than substantive. Toward the end of the sentence, *conois* is aligned with *fes*. In this case, their p and m properties are equal (p = verb, m = present first person singular), but the l properties carry different values, which means that the variant is lexical. 56

Witness	**Tokens**									
A		ge	te	conois	mielz	que	tu	ne	conois	moi
B	Artu	je	te	conois	miels	que	tu	ne	fes	moi
Variant type		form.			form.				lex.	

Table 19. Categorized variants

This information could be passed along to the **Visualization** stage for different purposes. For example, the output of the **Visualization** stage might be a critical apparatus where variants are grouped according to whether they agree in substantive properties, regardless of whether their n values coincide. Independently of the way variants might be presented in a text intended for critical reading, tables or graphic representations of the pairwise agreement in different types of significant features among witnesses might let us hypothesize about textual filiation and transmission. From this perspective, the output in the **Visualization** stage might include a report about textual relationships among the witnesses, or a dendrogram representation of the result of agglomerative clustering of the witnesses, as in a phylogenetic tree. Such an analytic product is no less a visualization of textual relations than a reading view that reproduces the full text with apparatus, even though it is an abstraction of the relationships, and not an organized reproduction of the primary data. Most importantly, as long as the pipeline is completely automated, different scripts can be run or re-run in order to obtain different, but complementary visualizations. 57

Stage 5. Visualization

CollateX supports several output views of the variant graph, including an alignment table (in plain text or HTML, as well as CSV and TSV), a graph (in SVG), XML (both TEI and a generic XML that is suitable for subsequent transformation with XSLT), and a JSON document. Of these built-in output views, only the JSON output renders both the n property and all custom properties assigned during earlier processing, which means that users can post-process the JSON output in order to create custom visualizations. Some other built-in output formats are also able to render more than one property value for a token. For example, two types of SVG output are supported, one with just the n property value and the other with the n value and, for each node in the variant graph, all associated t values and the identifiers of the witnesses in which those readings are found, as in the figure below (Fig. 2). In this graphic, the n value is made of the part of speech and the lemma, and is written in bold in the top left cell for each node; the corresponding t values are listed below, together with the identifiers of the witnesses in which they can be found. 58

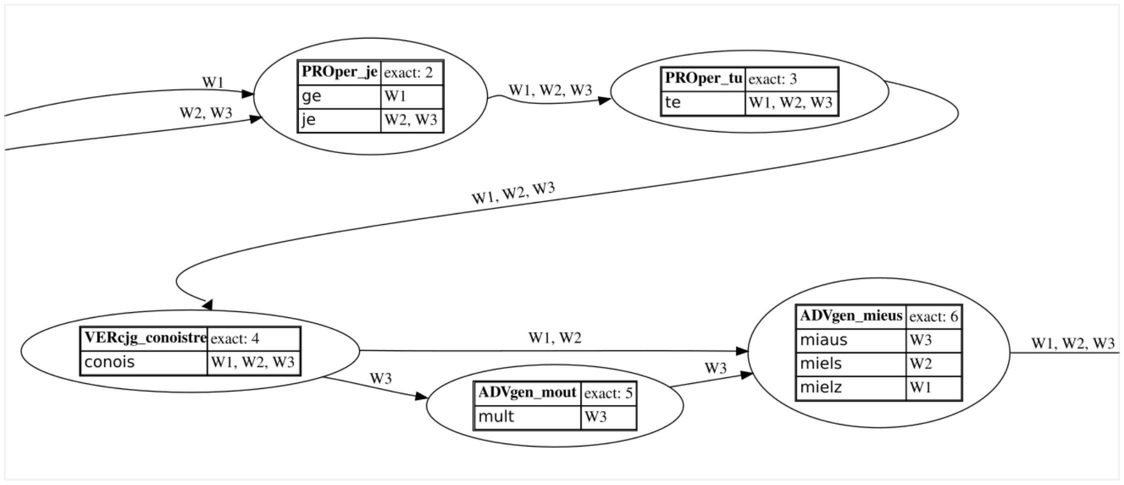

Figure 2. CollateX output SVG graph.

Conclusion

Revisiting the Gothenburg model and its CollateX implementation

The Gothenburg model has advanced our understanding of the machine-assisted analysis of textual variation in both conceptual and practical ways. The modular nature of the model recognizes both the substantial independence of the five stages and, at the same time, the extent to which they interact within a processing pipeline. This modular conceptualization, in turn, has enabled modular implementation, where an application such as CollateX incorporates hooks that permit the user to modify one stage of the pipeline without having to interact with the others except by continuing to observe the input and output specifications built into the API. 59

Meanwhile, with more than ten years of experience of the Gothenburg model behind us, we now also recognize aspects of the model, and of its implementations, that could benefit from further consideration. For example, **Visualization** (Stage 5) might more transparently be called “Output” or “Rendering.” The implementation of the model in CollateX is a good example of this, since the richest output format supported, JSON, is intended not for human visual consumption, but for automated post-processing. Additionally, in CollateX the modular stages might have been implemented with even greater mutual independence. For example, the output format (text table, HTML, SVG, JSON, etc.) is specified as an argument to the function that performs the **Alignment**, which means that generating multiple output formats requires performing the same **Alignment** operation anew for each of them.[32] The clearer understanding of the model that comes from a decade of experience with the CollateX implementation suggests directions for further development that could continue to enhance the benefits that are already visible in both the modular nature of the Gothenburg model and its implementation in the software. 60

Revisiting normalization

A uniquely broad and consequential insight in this retrospective context is that **Normalization** (Stage 2) and normalization (with a lower-case “n”) are different, and normalization (small “n”) is broadly and correctly distributed over several stages of the pipeline. **Normalization** (Stage 2) means identifying string-level differences in witness tokens that should be ignored for alignment purposes, but the point of this report has been to explore the many ways in which normalization (small “n”) is pervasive, affecting not only Stage 2, but also Stage 0 (**Transcription**), Stage 1 (**Tokenization**), and Stage 4 (**Analysis**). The type of normalization that is applied at these different stages emerges from a variety of considerations. For example, the character-based transcription of handwritten sources in Stage 0 necessarily entails normalization because handwriting is infinitely variable, and infinite variation cannot be represented usefully in character data. And the normalization applied in Stage 4 to support near-matching in CollateX is an accommodation to the intractable computational complexity of implementing near-matching globally as part of the **Alignment** stage. In other words, despite the overall clarity, accuracy, and utility of the modularization of the 61

Gothenburg model, both scholars and developers benefit from an awareness that normalization as part of a collation pipeline neither begins nor ends with Stage 2, and also that normalization may be used in analysis and reporting in situations that are independent of alignment.

Figure 3, below, reproduces Figure 1, the earlier plot of the five stages of the Gothenburg model of textual collation, with the addition of an explicit Transcription (or digitization) stage between the Input (the documentary artifact) and **Tokenization**, which serves as Stage 1 of the original model. Additionally, in this revised figure *all* of the stages that involve normalization are highlighted in green: 62

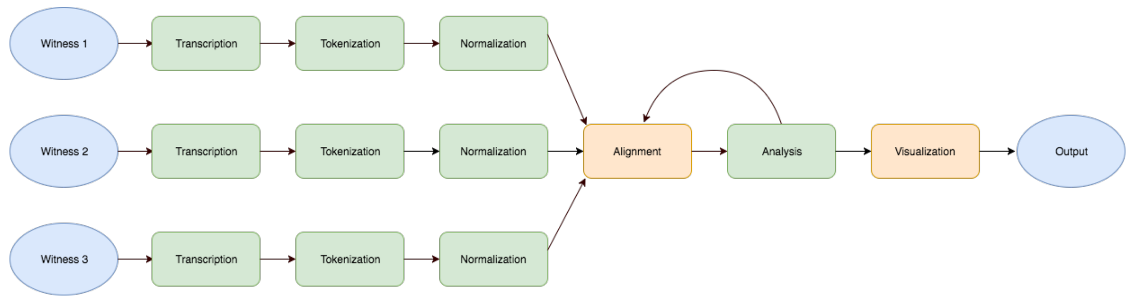

Figure 3. Gothenburg model (with Transcription)

This figure represents a pipeline, corresponding to an editorial workflow. Despite our identification in this report of a broad distribution of normalization operations beyond the titular **Normalization** Stage, the modular pipeline architecture, including that stage, remains fundamental both as a way of modeling the collation process and as a guide for implementation. Indeed, the existence of an official **Normalization** Stage between **Tokenization** and **Alignment**, where differences among witness tokens are neutralized in order to enable their comparison for alignment purposes, provides a context for recognizing and understanding the aspects of normalization that necessarily take place elsewhere. 63

Notes

[1] We capitalize and embold **Normalization** when we refer to the second step in the Gothenburg model. When we refer to the general concept of normalization (that is, of removing what the researcher considers insignificant variation during the editorial process), independently of the Gothenburg model, we render the word without special capitalization or other formatting.

[2] In the most common case, the entire output of one process becomes the entire input into the next, although Unix piping also supports tee connections, which send the same output to two destinations, typically stdout and a specified file or variable.

[3] The dotted line indicates that Stage 4 (**Analysis**) may pipe its output into another Stage 3 (**Alignment**) process, representing a situation where **Analysis** may require a new **Alignment**, which may or may not be subjected to additional **Analysis** (or additional cycles of **Analysis** and **Alignment**) before visualization. As Haentjens Dekker and Middell (2011) write:

> an additional analytical step in which the alignment results are augmented (and optionally fed back as preconditions into the collator) appears essential to us in order to bridge the methodological “impedance” between a plain computational approach and the established hermeneutical “best-practice” approach to the problem.

[4] The adoption of the design principle called *separation of concerns* in the field of automatic collation dates back to the 1970s. See, e.g., Gilbert (1973), 144: “The program is modular in design, that is, it consists of several steps each doing a simple task.” The importance of this modularity in tool design in general, and not only for collation, is highlighted also in Rockwell and Bradley (1999), section 1.1:

> The unforeseeable nature of original research means that a research tool cannot be extended in a predictable way to meet the demands of research. What is needed are research tools whose every aspect can [be] modified and extended through easily added modules. In other words, to be truly extensible, a tool should be capable of having any component replaced.

[5] The Gothenburg model originated in the context of Interedition, a European-funded collaborative research project whose aim was “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” [Interedition]

[6] Our examples are drawn from the Python version of the software, which is available at https://pypi.python.org/pypi/collatex; the Java version

is available at https://collatex.net. See also the discussion of the implementation of the Gothenburg model in Juxta at http://juxtacommons.org/tech_info.

[7] Especially in the early days of computers, hardware and software constraints limited ways of representing digital texts; see, e.g., Froger (1968), 230–32. Conventions were quickly adopted to overcome these limitations, such as the use of the $ sign as a diacritic to indicate a capital letter.

[8] A manuscript is typically a multilayered reality [Sels and Birnbaum 2015], where various systems of language and meaning coexist [Segre 1976]. Useful tokenization and normalization demands careful attention to distinguishing orthographic and linguistic features of the text from those of the manuscript.

[9] http://www.tei-c.org/release/doc/tei-p5-doc/en/html/ref-orig.html

[10] Tokenization and normalization are well-known operations in computational and corpus linguistics, and their use in CollateX is similar to the analogous linguistic operations. Cf. tokenization in computer science, sometimes called *lexing* or *lexical analysis*, which shares with linguistic tokenization the goal of dividing a continuous stream of characters into meaningful substrings for subsequent processing. In computer science, the subsequent processing is typically *parsing*, that is, interpreting the tokens as expressions according to the syntax of a programming language.

[11] In Python regular expressions, `\s` “[m]atches any whitespace character; equivalent to `[ \t\n\r\f\v]` in bytes patterns or string patterns with the ASCII flag. In string patterns without the ASCII flag, it will match the whole range of Unicode whitespace characters” [Secret Labs 2001]. See Unicode Consortium (2020) for an inventory and discussion of Unicode whitespace characters.

[12] The default implementation of Tokenization in the Python version of CollateX is performed by the `tokenize()` method of the WordPunctuationTokenizer class inside `core_classes.py: re.findall(r'\w+\s*|\W+', contents)`. This defines a token as 1) either a sequence of word characters followed optionally by a sequence of whitespace characters, or 2) a sequence of non-word characters, which is typically punctuation, also followed optionally by a sequence of whitespace characters. This means that a token may consist entirely of whitespace only when it falls at the beginning of a text (where it will match `\W+`), since any whitespace not at the beginning of the text will form the trailing part of the token that begins with whatever word or punctuation characters precede it.

[13] Default tokenization in CollateX is described above. Default normalization is limited to stripping trailing whitespace in the Python version, and includes both that and case folding in the Java version. In the Python version, the normalization is created for objects of the Witness class (defined inside `core_classes.py`) with: `Token({'t': token_string, 'n': re.sub(r'\s+$', '', token_string)})`. The regular expression `\s+$` matches one or more whitespace characters at the end of the token; the `re.sub()` function replaces them with the empty string, that is, removes them from the token before writing the modified form as the value of the n property.

[14] Neutralizing phenomena that are not considered relevant for the purpose of alignment so that they would not add noise to the output was a concern also in the early days of automatic collation; see Nury and Spadini (2020), Silva and Love (1969), 93; Gilbert (1979), 247; Robinson (1989), 100–01.

[15] We use the term *forced match* to designate a situation where a single token is sandwiched between unambiguous matches of all witnesses on either side, as in the case-folding example immediately below. The match is forced because, with the neighbors fully aligned, the tokens between them, whether the same or different, are forced into alignment.

[16] See *Frankenstein variorum*. In the source files used in the project, this heading also includes presentational markup, which could be recruited to override case distinctions in the character data as part of a strategy for controlling normalization.

[17] See PVL. Cyrillic used letters of the alphabet as numerical digits, and these two letters continued to be distinguished in early Cyrillic when they represented numerical values even after any distinction between them had largely ceased to be significant in words.

[18] Normalization in support of alignment in this edition also implements case folding, neutralizes several other character distinctions (including “ѧ” vs “я” in this word), and ignores line breaks (represented by a vertical bar), punctuation, and superscription. The examples in this article simplify the actual CollateX output by representing characters tagged with `<sup>` tags as superscript characters and removing the `<sup>` tags and all others (principally `<lb>` and `<pb>`). This markup is removed from the n properties when they are created during Stage 2 **Normalization**, and therefore is not involved in Alignment decisions.

[19] **Alignment** in CollateX by default distinguishes only exact matches and non-matches, and it has no understanding of *near matching*, that is,

of finding the closest inexact match. In this case that means that the **Alignment** stage alone cannot recognize that “ge” is more similar to “je” than it is to “artu(s).”

[20] This would happen because, as noted above, in case of multiple alternatives regarded as equivalent by the software, CollateX pushes the token to the left.

[21] The researcher may also attach other properties, in addition to t and n, to tokens, and these can be returned as part of the output. Properties other than n are not used by the default alignment engine inside CollateX, but, in keeping with the modularity of the Gothenburg model, a researcher could replace the default CollateX alignment routine with an alternative that performs a more complex evaluation. For example, it is theoretically possible to perform several types of normalization, assign their results to different token properties, and perform the alignment in a way that assigns different weights to different degrees of matching in different token properties.

[22] Philologists may disagree about which of two possible alignments to prefer, and some such decisions have no clearly defined resolution. As a simplified example, given “very interesting” and “very, very interesting”, there is no clear way to decide whether to align the single instance of “very” in the first witness with the first matching instance in the second witness, or with the second matching instance.

[23] Repetition, when the same token appears multiple times in the witnesses, makes it difficult to determine which instances in one witness to align with which instances in another. The greater the extent of the repetition, the greater the risk of incorrect alignment.

[24] A forced match and an alignment based on shared n property values are represented differently in the variant graph that CollateX produces. That difference is not translated into the alignment table output, but it is part of the CollateX SVG rendering of the graph.

[25] In situations where researchers are able to hypothesize about the relationships among the witnesses before performing machine-assisted collation, they can exploit order dependencies of the algorithms by comparing the two witnesses that are assumed to be most closely related first. This approach will fail, though, in situations where different witnesses may be closest in different portions in the work. Additionally, and more generally, confidence about which witnesses are most closely related is proportional to the extent to which the collation has already been completed, a circular workflow that effectively amounts to requiring collation as a precursor to deciding how to implement the collation. CollateX has limited control over the order in which witnesses are incorporated into a collation. It can add witnesses one by one in an order specified by the researcher, but it cannot compare, for example, witnesses A and B and, separately, C and D, and then merge the two (A+B, C+D) interim variant graphs. That is, after comparing the first two witnesses and producing an initial variant graph, CollateX always aligns exactly one new witness at a time against the variant graph, and it cannot align a new witness directly against another new witness or an interim variant graph directly against another interim variant graph.

[26] For summary descriptions of the prevalent algorithms see https://en.wikipedia.org/wiki/Multiple_sequence_alignment.

[27] CollateX documentation is available at https://collatex.net/doc/#analysis-feedback.

[28] For a previous implementation of near matching, see Robinson (1989), 103.

[29] CollateX performs near matching not by revising the alignment table directly, but by adjusting the rank of the token in the variant graph, which, among other things, informs the layout of the alignment table. This makes the adjusted alignment information available at a lower level, and therefore also in output structures that do not use the alignment table, such as graph or SVG output.

[30] Another way to regard this issue emerges from the diasystemic nature of manuscript evidence, where (to simplify) features of the text are transmitted through the filter of features of the manuscript (that is, scribal practice) [Segre 1976]. During this transmission, the features traditionally considered significant (e.g., lexical) are those that are not affected subconsciously by scribal practice or habit as easily as those traditionally considered insignificant (e.g., orthography, although this, too, is a simplification). From this perspective, insofar as scribal norms or habits may be independent of the content of the text being copied, focusing on significant variation may be considered a strategy for prioritizing features of the text over those of the manuscript. Andrews 2016 challenges the traditional division of features into significant and insignificant, finding that, in the artificial traditions she examined, “human judgement was not significantly better than random selection for choosing the variant readings that fit the stemma in a text-genealogical pattern.” [Andrews 2016, 523]

[31] This approach is explored in Camps et al. 2019, for which see also https://github.com/CondorCompPhil/falcon.

[32] More strictly, performing the **Alignment** and generating the output are already separate pipeline steps within CollateX, but the API does not expose them individually.

Works Cited

Andrews 2016 Andrews, Tara L. 2016. “Analysis of variation significance in artificial traditions using Stemmaweb”. *Digital scholarship in the humanities* 31, no. 3 (2016): 523–39. https://doi.org/10.1093/llc/fqu072.

Andrews 2020 Andrews, Tara L. 2020. “Abusing the concept of normalization for better collation results (and profit)”. https://hcommons.org/deposits/item/hc:31925.

Birnbaum 2015 Birnbaum, David J. 2015. “CollateX normalization”. Presented at the “Computer-supported collation with CollateX” workshop, DH2015, Sydney. https://github.com/DiXiT-eu/collatex-tutorial/blob/master/unit7/soundex-normalization.pdf.

Birnbaum and Eckhoff 2018 Birnbaum, David J. and Hanne Martine Eckhoff. 2018. “Machine-assisted multilingual alignment of the *Codex Suprasliensis*”, in Stephen M. Dickey and Mark Richard Lauersdorf, eds, *V zeleni drželi zeleni breg. Studies in honor of Marc L. Greenberg*, 1–14. Bloomington, IN: Slavica.

Bleeker 2017 Bleeker, Elli. 2017. “Mapping invention in writing: digital Infrastructure and the role of the editor”. PhD diss., University of Antwerp. https://repository.uantwerpen.be/docman/irua/e959d6/155676.pdf.

Camps et al. 2019 Camps, Jean-Baptiste, Lucence Ing, and Elena Spadini. 2019. “Collating medieval vernacular texts. Aligning witnesses, classifying variants”. In *DH2019 Digital humanities conference 2019*. Utrecht, Netherlands. https://hal.archives-ouvertes.fr/hal-02268348.

CollateX Doc CollateX — software for collating textual sources. Documentation. https://collatex.net/doc/ (Java version) and https://github.com/interedition/collatex/blob/master/docs/pythonport.md (Python version).

Dekker and Middell 2011 Haentjens Dekker, Ronald and Gregor Middell. 2011. “Computer-supported collation with CollateX. Managing textual variance in an environment with varying requirements”. Paper presented at the meeting of Supporting Digital Humanities 2011, Copenhagen. In Bente Maegaard, ed., *Supporting digital humanities, Copenhagen 17–18 November 2011: conference proceedings.*

Dekker et al. 2015 Haentjens Dekker, Ronald, Dirk van Hulle, Gregor Middell, Vincent Neyt, and Joris van Zundert. 2015. “Computer-supported collation of modern manuscripts: CollateX and the Beckett Digital Manuscript Project”. *Literary and linguistic computing*, 30, no. 3 (1 September 2015): 452–70, https://doi-org.pitt.idm.oclc.org/10.1093/llc/fqu007.

FV n.d. *Frankenstein variorum.* https://frankensteinvariorum.github.io/viewer/.

Froger 1968 Froger, Jacques. 1968. *La critique des textes et son automatisation.*. Coll. “Initiation aux nouveautés de la science” n° 7. Paris, Dunod.

Gilbert 1973 Gilbert, Penny. 1973. “Automatic collation: a technique for medieval texts”. *Computers and the humanities* 7: 139–47. https://www.jstor.org/stable/30199534.

Gilbert 1979 Gilbert, Penny. 1979. “The preparation of prose-text editions with the ‘Collate’ System”. In *La Pratique des ordinateurs dans la critique des textes*, 245–54. Paris: Ed. du C. N. R. S.

Interedition Interedition. http://www.interedition.eu/.

McIlroy 1964 McIlroy, Douglas. 1964. “Summary — what's most important”. http://doc.cat-v.org/unix/pipes/.

Needleman and Wunsch 1970 Needleman, Saul B. and Christian D. Wunsch. 1970. “A general method applicable to the search for similarities in the amino acid sequence of two proteins”. *Journal of molecular biology* 48 (3), 443–53. https://doi.org/10.1016/0022-2836(70)90057-4.

Nury and Spadini 2020 Nury, Elisa, and Elena Spadini. 2020. “From giant despair to a new heaven: the early years of automatic collation”. *It - Information Technology* 62 (2): 61–73. https://doi.org/10.1515/itit-2019-0047.

PVL n.d. *PVL. Povest' vremennyx let.* http://pvl.obdurodon.org/.

Robinson 1989 Robinson, P. M. W. 1989. “The collation and textual criticism of Icelandic manuscripts (1): collation”. *Literary and linguistic computing* 4 (2): 99–105. https://doi.org/10.1093/llc/4.2.99.

Robinson and Solopova 1993 Robinson, Peter and Elizabeth Solopova. 1993. “Guidelines for transcription of the manuscripts of *The Wife of Bath’s prologue*”, in Norman Blake and Peter Robinson, eds, *The Canterbury Tales Project occasional papers* I, 19–51. Oxford: Office for Humanities Communication.

Rockwell and Bradley 1998 Rockwell, Geoffrey and John Bradley. 1998. “Eye-ConTact: towards a new design for text-analysis tools”. *Digital studies/Le Champ numérique*, February. https://doi.org/10.16995/dscn.232.

Secret Labs 2001 Secret Labs. 2001. “Secret Labs’ regular expression engine”. https://github.com/python/cpython/blob/3.6/Lib/re.py.

Segre 1976 Segre, Cesare. 1976. “Critique textuelle, théorie des ensembles et diasystème”. *Bulletin de la classe des lettres et des sciences morales et politiques de l’Académie Royale de Belgique* 62 (1976): 279–92. https://www.persee.fr/doc/barb_0001-4133_1976_num_62_1_55259.

Sels and Birnbaum 2015 Sels, Lara and David J. Birnbaum. 2015. “Editing the *Bdinski sbornik* as a multilayered reality”. In *Агиославика. Проблеми и подходи в изследването на Станиславовия чети-миней: доклади от едноименната конференция - 21 май 2013 г. (Hagioslavica. Issues and approaches in the study of the Stanislav Reading Menaion: presentations from the conference of May 21, 2013.)*, ed. Diana Atanasova. Sofia: Kliment Oxridski University, 2015 (appeared in May 2016), 184–99.

Silva and Love 1969 Silva, Georgette, and Harold Love. 1969. “The identification of text variants by Ccmputer”. *Information Storage and Retrieval* 5 (3): 89–108. https://doi.org/10.1016/0020-0271(69)90014-X.

Soundex Soundex. https://en.wikipedia.org/wiki/Soundex.

Spadini 2016 Spadini, Elena. 2016. “Studi sul ‘Lancelot en prose.’” PhD diss., Sapienza Università di Roma. http://hdl.handle.net/11573/1307347.

Spadini 2017 Spadini, Elena. 2017. “The role of the base manuscript in the collation of medieval texts”, in *Advances in digital scholarly editing. Papers presented at the DiXiT conferences in the Hague, Cologne, and Antwerp,* eds. Peter Boot, Anna Cappellotto, Wout Dillen, Franz Fischer, Aodhán Kelly, Andreas, Mertgens, Anna-Maria Sichani, Elena Spadini, and Dirk van Hulle. Leiden: Sidestone Press, pp. 345–49. https://www.sidestone.com/books/advances-in-digital-scholarly-editing.

Unicode Consortium 2020 Unicode Consortium. 2020. The Unicode standard version 13.0 – core specification. Chapter 23, “Special areas and format characters”, 881–916 (esp. 885-86). https://www.unicode.org/versions/Unicode13.0.0/ch23.pdf.