Revisiting multiple-witness text alignment

Ronald Haentjens Dekker, Humanities Cluster, KNAW, ronald.dekker@di.huc.knaw.nl
David J. Birnbaum, University of Pittsburgh, djbpitt@gmail.com

Introduction

Philologists often need to reconcile multiple copies of a work that share readings in some places and diverge in others. Philologists care about aligning the text of these copies (called *witnesses*) because patterns of agreement and variation provide insight into how the text changed during copying over time.[1]

Current approaches

Aligning multiple sequences, such as multiple textual witnesses to a work, is known to be an NP-hard problem, that is, a task that, for more than a small amount of data, requires heuristic compromises in order to yield acceptable performance. As an illustration of these sorts of compromises we can consider CollateX, an open-source application that performs textual alignment for multiple witnesses, and that is in wide use in computational philology (e.g., in StemmaWeb[2], University of Münster Institute for New Testament Research[3], Textual Communities[4]). The current Java and Python release versions of CollateX perform *progressive alignment*; that is, when aligning multiple witnesses, they begin by merging two witnesses into a partially ordered *superwitness*, and they then fold one full additional witness at a time into the alignment. The internal data model used by the current release versions of CollateX to manage this process is the *Rhine Delta* (also known as the *variant*

[1] As we write in Birnbaum and Haentjens Dekker 2024 (fn 1), "sequence alignment is also an important task in bioinformatics, although the contexts for bioinformatic and philological sequence alignment differ. See the discussion at Birnbaum 2020 §1.2."

[2] https://stemmaweb.net/

[3] https://www.uni-muenster.de/INTF/en/

[4] https://textualcommunities.org/app/

graph[5]), a directed acyclic graph structure that provides a compact representation of patterns of agreement and variation within the witness set.

Issues with current approaches

The workflow in the current CollateX release versions raises the following issues:

1. Although the current Java and Python release versions of CollateX employ different alignment algorithms, both become impractically slow when aligning long witnesses.

2. The progressive alignment of full witnesses does not permit backtracking, which means that a suboptimal choice early in the process, which cannot be undone, impinges on the quality of the final result.[6]

3. Adding information incrementally to a Rhine Delta structure, as in the current release versions of CollateX, requires performing a new topological ordering of the entire graph after each modification. A more efficient method would reduce the need for topological ordering operations.

Features of the new algorithm

The new algorithm operates in two phases (described in more detail below):

1. A first phase aligns *full-depth* (that is, all-witness) moments of agreement within the witnesses, commits those to the result, and then is applied recursively to the spaces between them until no further full-depth alignments can be found.

2. A second phase, applied to areas that cannot be fully aligned by the first phase, employs an enhanced progressive-alignment strategy to identify and resolve remaining, non-full-depth alignments.

---

[5] The term *variant graph* is common in discussion of textual alignment, including in the current CollateX documentation (see CollateX), but the term *Rhine Delta* (introduced in Sperberg-McQueen 1989) is more appropriate because it predates the term *variant graph* (introduced in Schmidt and Colomb 2009) by two decades. The model used in current release versions of CollateX is also closer to the original Rhine Delta description than it is to the 2009 variant-graph description. A complete history of the model and associated terminology can be found in the "Graphic visualizations" section of Birnbaum and Haentjens Dekker 2024.

[6] See Spadini 2017: 247–48. As we discuss in Birnbaum and Haentjens Dekker 2024 (fn 9), *iterative* alignment, which begins with progressive alignment and then, once more information is available, removes and reincorporates individual witnesses, can correct some, but not all, early suboptimal alignment decisions.

The new alignment algorithm improves on the methods employed in the current release versions of CollateX in the following ways:

Improved performance

A first phase begins by identifying all full-depth alignments of tokens; we call these alignment areas *full-depth blocks*. Because repeated blocks require additional alignment decisions, we remove repeated blocks from consideration (temporarily), and we also remove embedded blocks, that is, blocks that are fully contained by other blocks. We then employ a beam search to traverse a graph of the remaining blocks in a non-greedy way, leaving multiple options open while collecting additional information in order to select an initial result that aligns the largest number of tokens with the fewest path steps.

The alignment of full-depth blocks described above partitions the space, and we then reapply the first-phase method recursively on the unaligned areas (which we call *islands*) between the full-depth blocks whose alignment we have resolved. The partitioning often resolves what initially were instances of repetition and transposition, and it also broadens the definition of full-depth, which, after the first pass, comes to refer to all witnesses present in an island, even when that is not all witnesses present in the entire corpus. These resolutions mean that blocks that we could not align initially might be eligible for alignment during this recursive application of the first-phase strategy. A key feature of this approach is that full-depth alignments are *qualitatively* different even from *almost*-full-depth alignments because only full-depth alignments are able to partition the space.

We conclude this recursive first phase only when there are no remaining unaligned, non-repeating full-depth blocks.

Improved implementation of progressive alignment

During a second phase we employ a heuristic clustering of the readings present in an unaligned island to construct a *guide tree,* that is, a hierarchical representation of the order in which individual witnesses should be merged into the alignment. We then apply progressive alignment according to the clustering result, which means that we traverse the clustering hierarchy from the bottom up and merge, at various moments, the two locations in the hierarchy that have the greatest similarity. This means that at various times we merge two single witnesses, a single witness and a partially ordered graph representing an interim alignment, and—crucially—two partially ordered graphs representing interim alignments.

The implementation of progressive alignment in the second phase of the new algorithm improves on the current release versions of CollateX, increasing the likelihood of adding witnesses to the alignment in an optimal order, in three ways:

1. We use progressive alignment only when no further order-independent alignment, according to the first phase, is possible.

2. We construct the guide tree locally, over a small space. This increases the likelihood of selecting an order that may be optimal within an island, even when it is not necessarily optimal globally.

3. We incorporate the ability to merge two graphs when those are the nodes in the clustering hierarchy that are closest at that point in the process. This improves over the simpler progressive model used in the current release versions of CollateX, which is able to merge only two single witnesses or one single witness plus one graph, but not two graphs.

Reduced topological ordering

The new alignment algorithm reduces the impact of topological ordering on the amount of effort in the following ways:

1. We perform topological ordering only during the second phase, and therefore not for all tokens. This means that while no token gets added to the result in the current release versions of CollateX without participating in topological ordering, the new algorithm employs topological ordering only to text that gets incorporated during the second phase and only within individual islands.

2. The new hyperedge method constructs an unordered hypergraph and infers an order only when the hypergraph is complete and only for the nodes that are included in potential matches. The Rhine Delta method implements a directed ordered graph and therefore maintains and updates the order of every node after every merge in the hierarchical clustering.

3. The current release versions of CollateX build the Rhine Delta structure out of individual tokens, while the new algorithm operates with ranges of adjacent tokens. This means that any topological ordering is applied to the ranges, and not to the individual tokens, thereby reducing the number of nodes to be ordered.

We can visualize the hypergraph directly using an innovative visualization that we call an *alignment ribbon*.[7] Alternatively, we can convert the hypergraph to any of the visualization methods currently supported by CollateX, including the Rhine Delta, which remains available as a representation of the alignment result even though it is no longer used as the internal model during construction.

Conclusion

The alignment algorithm described here introduces the following new features:

1. The *alignment hypergraph* supports simpler and more efficient computational methods than the *Rhine Delta,* which it replaces as the internal alignment representation of an alignment.

2. The recursive subdivision of the sequences to be aligned during the first phase of the new algorithm improves the efficiency of the process by dividing a large problem into multiple small problems.

3. The recursive aspect of the first phase of the new algorithm resolves many transposition and repetition issues.

4. The new algorithm aligns hyperedges (representing ranges of sequential tokens), rather than individual tokens, reducing the number of units to be aligned.[8]

5. Progressive alignment is applied only in small (*island*) domains, and not over the entire space. This has two advantages:

    a. The optimal order for incorporating witnesses into the alignment may be different at different locations, and a local, rather than global, approach to witness ordering increases the sensitivity of the method to potentially divergent local conditions.

[7] Birnbaum and Haentjens Dekker 2024 introduces the alignment ribbon visualization and discusses the alignment algorithm only as much as is necessary to provide context for understanding the visualization. The present report takes the opposite perspective; it focuses not on the alignment-ribbon visualization, but on details of the underlying alignment algorithm.

[8] The early construction of token ranges means that these ranges sometimes need to be separated during the second phase, and that separation entails a processing cost. At the same time, because the ranges are relatively small, the cost of managing the separation operations does not outweigh the benefit of reducing the amount of processing elsewhere.

   b. The new method reduces the total effort expended on topological ordering operations because ordering small ranges is less expensive than ordering a long range that includes all of the shorter ones and more.

6. The version of progressive alignment employed in the new algorithm is able to merge two graphs, and not only two single witnesses or one single witness plus one graph. This makes the order in which witnesses are incorporated into the result more sensitive and responsive to witness similarity.

The new algorithm is slated for incorporation into the next major release of CollateX.

Works cited

- **Birnbaum 2020.** Birnbaum, David J. 2020. "Sequence alignment in XSLT 3.0." *XML Prague 2020 conference proceedings*, pp. 45–65. https://archive.xmlprague.cz/2020/files/xmlprague-2020-proceedings.pdf. Accessed 2024-12-11.

- **Birnbaum and Haentjens Dekker 2024.** Birnbaum, David J., and Ronald Haentjens Dekker. "Visualizing textual collation: exploring structured representations of textual alignment." Presented at Balisage: The Markup Conference 2024, Washington, DC, July 29 - August 2, 2024. In *Proceedings of Balisage: The Markup Conference 2024. Balisage Series on Markup Technologies*, vol. 29 (2024). https://doi.org/10.4242/BalisageVol29.Birnbaum01. Accessed 2024-12-11.

- **CollateX.** CollateX – Software for Collating Textual Sources. https://collatex.net/doc/. Accessed 2024-12-11.

- **Schmidt and Columb 2009.** Schmidt, Desmond and Robert Colomb. 2009. "A data structure for representing multi-version texts online." *International journal of human-computer studies*, v. 67, no. 6, pp. 497–514. doi:https://doi.org/10.1016/j.ijhcs.2009.02.001. Accessed 2024-12-11.

- **Spadini 2017.** "The role of the base manuscript in the collation of medieval texts." *Advances in digital scholarly editing*, Sidestone Press, Leiden, 2017, pp. 345–49.

- **Sperberg-McQueen 1989.** Sperberg-McQueen, C. M. 1989. "A directed-graph data structure for text manipulation." Abstract of a talk given at the conference The Dynamic Text, Ninth International Conference on Computers and the Humanities (ICCH) and Sixteenth International Association for Literary and Linguistic Computing (ALLC) Conference, at the University of Toronto, June 1989. http://cmsmcq.com/1989/rhine-delta-abstract.html, Accessed 2024-12-11.