Collation

NEH Institute 2022

Ronald Haentjens Dekker
14-07-2022

What is collation?

Compare multiple textual witnesses of a work against each other

Finds the differences and agreements between witnesses

Variation says something about how the witnesses are related

Input: Example Use Case

W1: The red and the black cat.

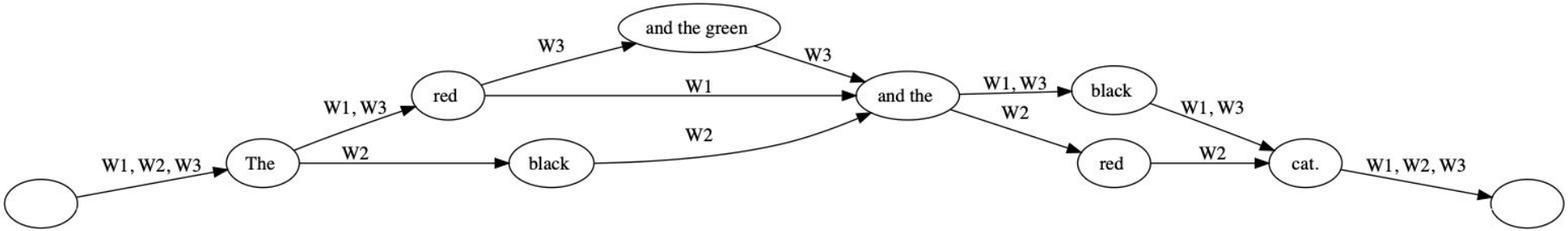
W2: The black and the red cat.

W3: The red and the green and the black cat.

Output: Alignment Table Visualisation

W1	The	red		and the	black	cat.
W2	The	black		and the	red	cat.
W3	The	red	and the green	and the	black	cat.

Output: Variant Graph Visualization



Second example

W1: The quick brown fox jumped over the lazy dog.

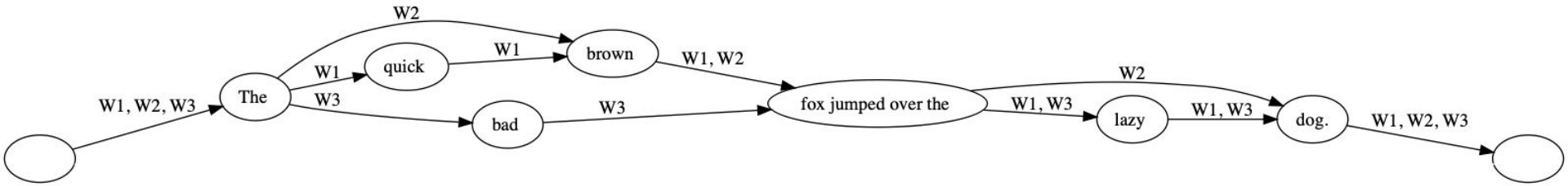
W2: The brown fox jumped over the dog.

W3: The bad fox jumped over the lazy dog.

Output Alignment Table Second Example

W1	The	quick	brown	fox jumped over the	lazy	dog.
W2	The		brown	fox jumped over the		dog.
W3	The	bad		fox jumped over the	lazy	dog.

Output Variant Graph Second Example



Requirements for a collation tool

Must work with different languages, ancient and modern.

Must be able to collate two or more witnesses

Must be able to work without selecting a base witness

Must be able to be integrated in different edition projects

Must be able to work with different input and output formats

Must be able to work with different programming languages and OSes.

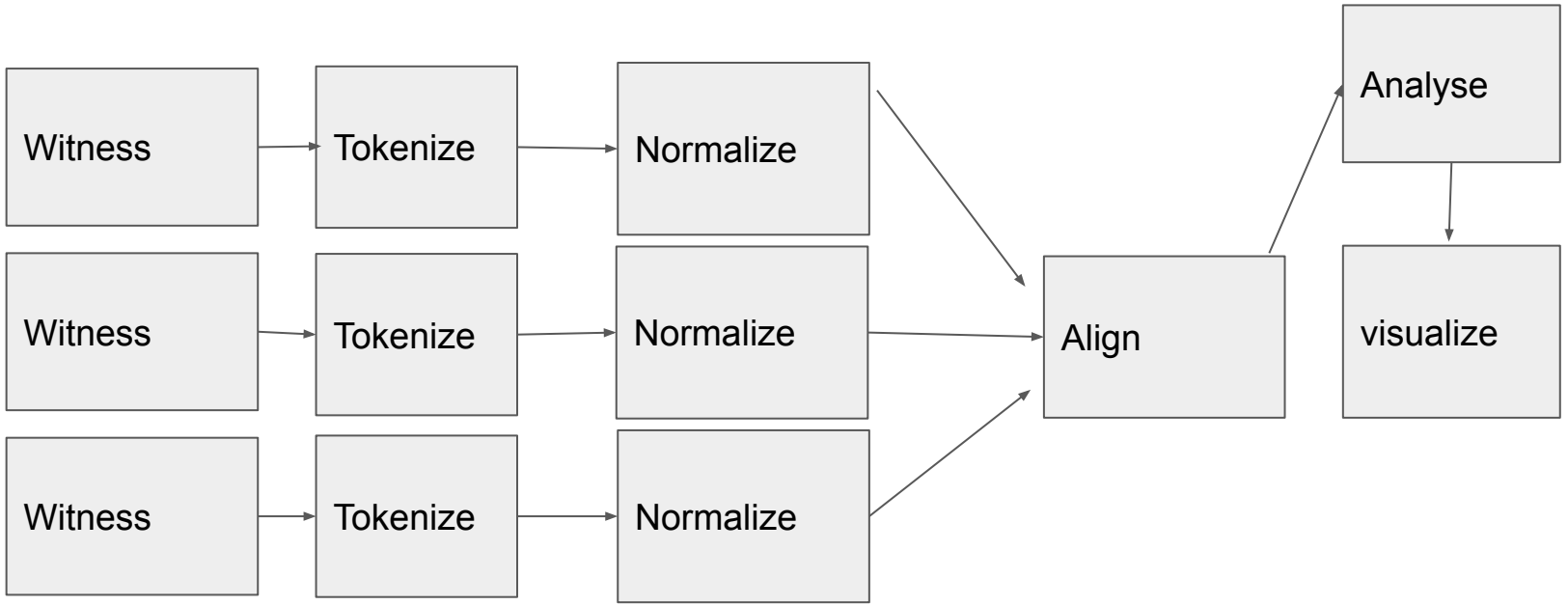
How do we accomplish that?

We divide the collation problem up into subproblems:

- Tokenization
- Normalization
- Alignment
- Analysis
- Export/Visualization

These steps together form a computational pipeline

Collation Computational Pipeline



General Approach

Tokenize, normalize each of the witnesses.

Align the witnesses.

Create graph structure internally.

Visualize the result.

Tokenize

- Split on whitespace and punctuation
- Deal with input format specific syntax

Normalize

- Change Uppercase or mixed case tokens into lowercase
- Strip whitespace
- Normalize numbers, i.e three becomes 3 so that a token “3” and “three” are normalized the same way
- Soundex normalization

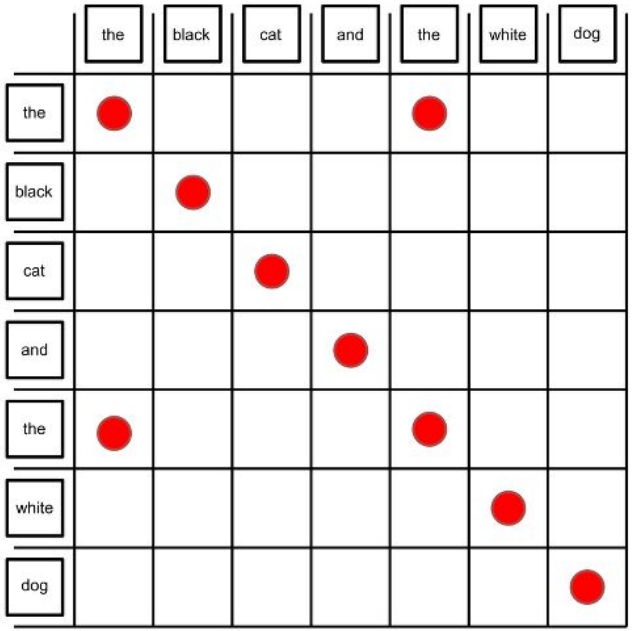
Pairwise alignment

Multiple well known algorithms:

- Longest common subsequence
- Dynamic programming, global alignment: Needleman-Wunsch
- Dynamic programming, local alignment: Smith-Waterman
- Global alignment with affine gap scoring: Gotoh

All edit operation based.

Alignment represented as a matrix

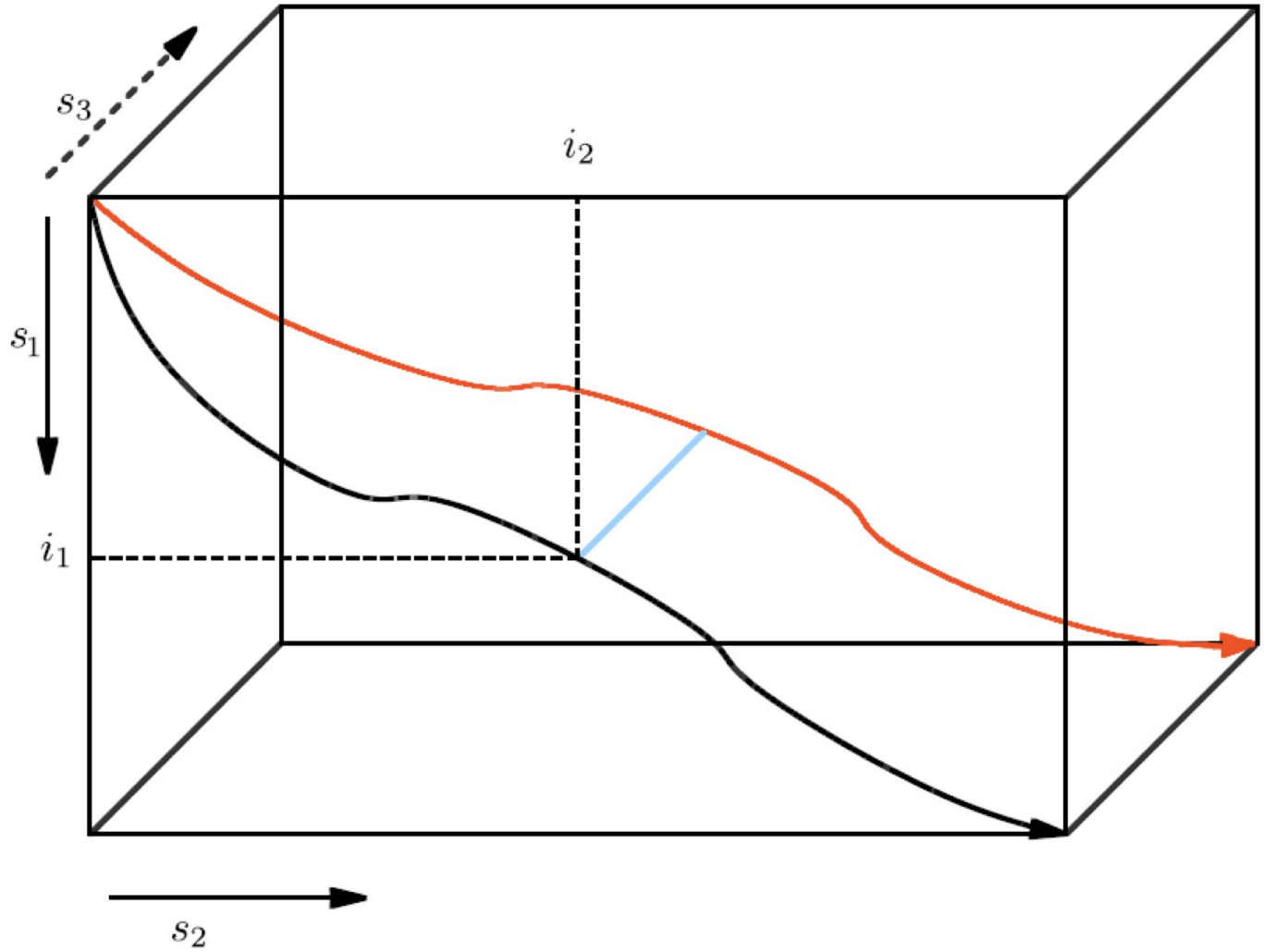


Needleman-Wunsch

match = 1 mismatch = -1 gap = -1

		G	C	A	T	G	C	U
	0	-1	-2	-3	-4	-5	-6	-7
G	-1	1	0	-1	-2	-3	-4	-5
A	-2	0	0	1	0	-1	-2	-3
T	-3	-1	-1	0	2	1	0	-1
T	-4	-2	-2	-1	1	1	0	-1
A	-5	-3	-3	-1	0	0	0	-1
C	-6	-4	-2	-2	-1	-1	1	0
A	-7	-5	-3	-1	-2	-2	0	0

Aligning 3 witnesses with dynamic programming.



Progressive alignment

Takes pairwise algorithm

Aligns the first two witnesses

Merges add operations and dels and matches in an artificial witness

Align third witness against result of the merge.

Repeat for forth, fith, .., n-witness.

Good performance

Supersequence is fictional

Introduces bias

Order of alignment matters. Can be user supplied order or can be based on guide tree using a heuristic.

Analysis

Try to improve the alignment by looking at

- Near matches in the variation
- Transpositions
- Manual corrections

Visualization / Export

- Alignment Table as plain text or HTML
- Variant Graph as SVG
- XML export
  - TEI parallel segmentation
  - GraphML output
- JSON output
  - Can be processed with JavaScript in the browser

CollateX

Open source textual collation engine

Implements the computational collation pipeline described earlier

Started as a prototype during Interedition project 2008-2012

Is able to collate multiple witnesses against each of other without selecting a base text

- Java version can be integrated in digital edition as a web-service
- Python version runs in Jupyter-Lab

XML and collation; challenges

Markup are annotations on text that add interpretation to text

Collation is an operation on text

1) How to deal with the markup?

XML/TEI files can contain adds and dels to indicate revisions in manuscripts, i.e. non-linear text.

2) How to collate text with internal variation?

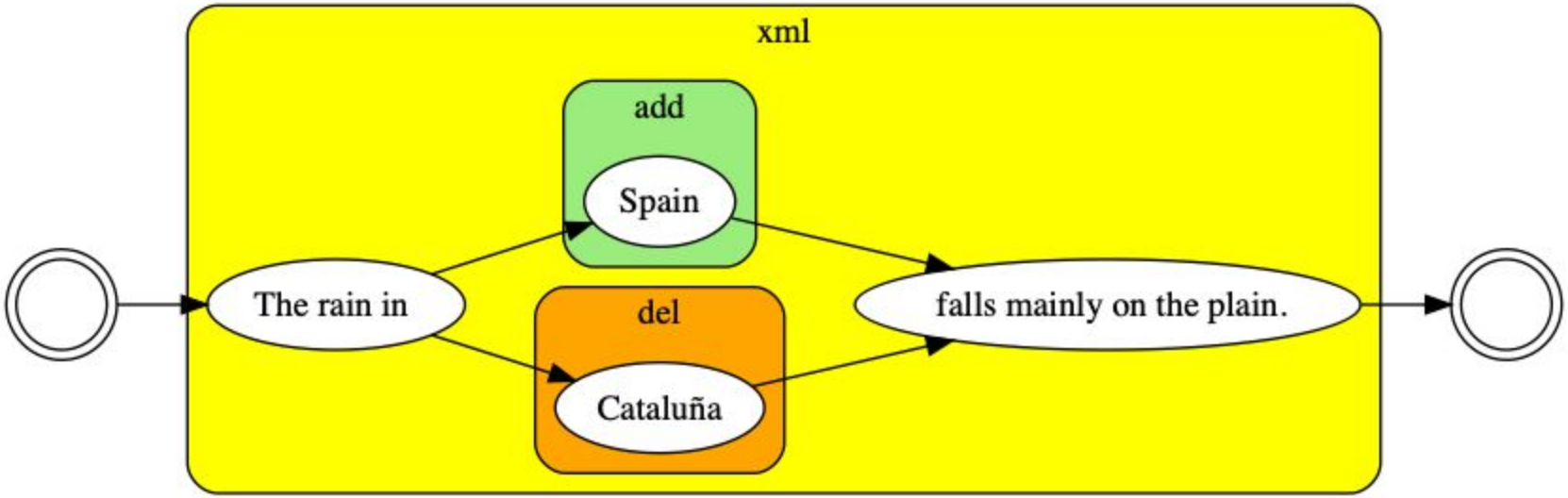
1. Collating XML files

- We can strip away all the markup, headers, metadata using an XML tokenizer and pass the tokens to the collation tool. The tokens contain position information so that the output can be linked to the input.
- We can use the markup to segment the textual content. For example: segment on Page, Chapter or Paragraph elements.
- We can pass along the markup as properties on the tokens. We can then use that extra information during the alignment to influence the matching phase of the alignment. We can also use the information during the visualisation stage by visaling tokens with certain type of markup on it differently.
- Native XML tree collation. Presents a problem. Which hierarchy is leading: the markup or the textual content? TEI/XML files are mixed content. So the answer is: it depends.

2. Collating nonlinear XML/TEI files

- We can flatten the textual witnesses by taking only the first version of the text by ignoring any deletions and additions, only applying instant corrections.
- Likewise we can flatten the textual witnesses by taking only the last version of the text by applying any and all deletions and additions.
- We can model each witness as a variant graph itself instead of a linear array of tokens. Alignment then becomes a problem of merging graphs.

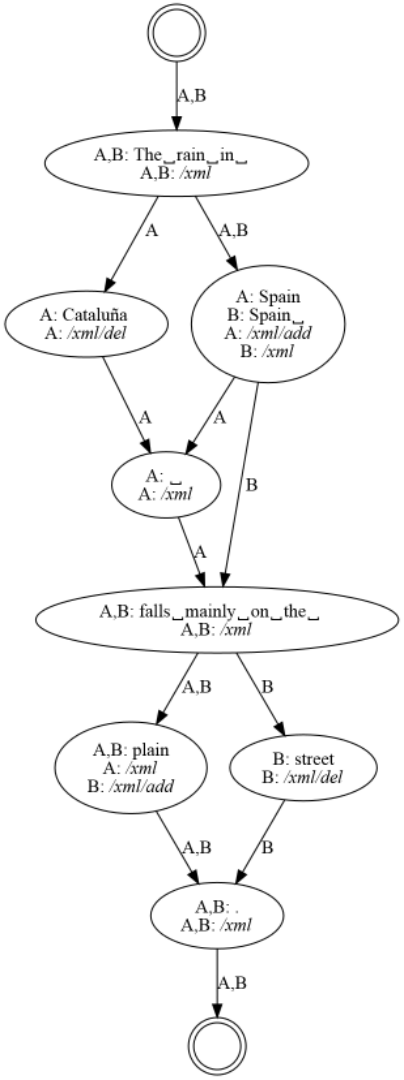
Treating a textual witness as a graph



Collating nonlinear witnesses results in a extended alignment table

[A]	The_rain_in_	[+] Spain [−] Cataluña	_	falls_mainly_on_the_	plain	.
[B]	The_rain_in_	Spain_		falls_mainly_on_the_	[+] plain [−] street	.

Collating nonlinear witnesses results in a variant graph with extra information on the nodes



Resources

Java version CollateX can be found at www.collatex.net

Python version CollateX can be found at https://pypi.org/project/collatex/

Elaborate installation instruction for Python version of CollateX: https://nbviewer.org/github/DiXiT-eu/collatex-tutorial/blob/master/unit1/Installation.ipynb

Hyper-Collate: https://github.com/HuygensING/hyper-collate