BOOK CULTURES

Further Information and Publications
www.arc-humanities.org/series/book-series/

NETWORK ANALYSIS FOR BOOK HISTORIANS

DIGITAL LABOUR AND DATA VISUALIZATION TECHNIQUES

by

LIZ FISCHER

ARC HUMANITIES PRESS

HOW TO NETWORK

THROUGH MY WORK with the datasets in this project, and with other datasets in other network analysis projects, I have come to a system of processes for taking data from analogue sources to digital networks. The print-to-network data processing pipeline has the following steps:

- **Plan** – Determine the target format or structure.
- **Model** – Identify the structure of the data and decide on a data model.
- **Retrieve** – Obtain machine-readable text.
  For printed text, that means:
  - **Scan** the print pages into a PDF.
  - **OCR** the PDF to create machine-readable text.
- **Segment** – Break the text into meaningful units, and assign an identifier to each.
- **Annotate** – Mark up each segment to capture the relevant metadata, nodes, and edges contained therein.
- **Reshape** – Regularize and reformat the data. Structure it according to the requirements of network analysis tools.
- **Analyze** – Explore the data using statistical methods or visualization.

In the rest of this chapter, I dive deep into each of these steps for processing data into a machine-actionable, network visualization-ready format and discuss the considerations required at each step.

Planning

Although analysis is the *last* step of the process of working with data, it is usually helpful to begin by thinking about the kinds of analysis one hopes to do and the relevant tools available, before embarking on the long journey of data processing—in order to plot a route, one must know the desired destination. This step can be thought of as "gathering requirements"—a process determining the needs and expectations of whatever tools and methods will

Network analysis requires one to identify two basic categories of "thing" in the data: nodes (the people/places/ideas/things of interest) and edges (the relationships that connect those things). To enable *interesting* analysis and visualization, one needs to know not just what each of these nodes and edges are but what properties they have. In general, network analysis requires one to gather:

- A list of each "object of interest," each thing that will be a node in the network—whether that be a book, a text, a person, etc. Each of these also needs a unique identifier.
- Metadata about each object of interest—what characteristics might one want to use as a filter to include or exclude objects from analysis later on? For example, if the object of interest is a text, it may be important to capture the author(s) name(s). If the object of interest is a book, the year of publication may be relevant.
- A list of relationships between these objects of interest—the edges of the network.
- Metadata about each relationship. What properties of the *relationships* might be useful as filters, or to know more about? For example, if modelling a network of correspondence, each edge might have a date.

In addition to determining what information to capture, one must determine what *format* that information needs to be in—how it should be encoded so that it can be processed by networking tools. The most popular tools for working with networks in a humanities context are NetworkX and Gephi.[1] NetworkX is a Python package appropriate for those who want to work programmatically with their data, especially if focused more on statistical measures. Gephi is a desktop application focused primarily on visualization—though it does include tools for calculating a variety of statistics—that is appropriate for those who want or need a lower-code approach to network visualization. Palladio—a browser-based application developed by Stanford's Humanities + Design Lab[2]—also makes occasional appearances in DH + Networks literature, as it was designed specifically with humanists in mind. While Palladio does offer a simpler interface than Gephi and integrated mapping functionality, its network visualization

1 *NetworkX*, https://networkx.org/; *Gephi*, https://gephi.org/.

feature set is severely limited in comparison. For datasets that are very large—much larger than what I am working with in this project—using NetworkX could be important, because Gephi can be slow on very large data. But because I am most interested in the visualization aspects, and because my datasets are not overwhelmingly large, Gephi is the tool I targeted in my data processing.

Gephi supports a variety of input types, but I have found the easiest to work with is the Spreadsheet, a variation on the CSV input. Spreadsheet inputs consist of two tables—one "nodes" table containing the identifiers and attributes (i.e., metadata) for all nodes in the network, and one "edges" table containing the identifiers for both nodes the edge connects as well as any attributes ascribable to the edge itself. For example:[3]

Table 1. Example of Gephi Spreadsheet input format

NODE TABLE	
ID	*Label*
A	"Node A"
B	"Node B"
C	"Node C"

EDGE TABLE		
Source	*Target*	*Label*
A	B	"Edge from A to B"
C	A	"Edge from C to A"

This target data format informs decisions made during the data preparation process. Next, for each data set, one needs to determine what sets of nodes and edges to produce based on the starting data and desired output format, as this will shape the kinds of visualization that will be possible. This process is called "data modelling."

3 Example from Gephi documentation, https://gephi.org/users/supported-graph-

Data Modelling

Humans are natural pattern-seekers that excel at making meaning from, and finding form in, blobs of information. For example, imagine you encounter a handwritten note that reads:

> Sir
> I received your very welcome letter, whereby I find you abundant in courtesies of all natures... I have sent up two of your books which have much pleasured me... I shall much long to see you at this place, where you shall command the heart of your affectionate friend and servant,
> Edward Dering
> Dover Castle
> May 10 1630[4]

Assuming you are familiar with the genre of a letter, you will immediately recognize this as one. You will also know, based on your internalized knowledge of the form, some key pieces of information about this letter—who it is from, where it was written, when it was written, and a vague sense of whom it was to. This process of taking what we know about the structure of information, often intuitively or subconsciously, and formalizing it is called "data modelling."

A data model is a representation of something that can be used (at least for some tasks) as a stand-in for the thing itself. It is an abstraction, not a copy; it is focused on the structure, not the content. Models always focus on some features and disregard others, and the decisions made about that focus when designing the model will change what kinds of questions the model can be used to answer. For example, a very simple model of a letter might consist of three attributes: addressee, content, and sender (Table 2).

Table 2. Simple letter data model

Addressee	Content	Sender
Sir	I received your very welcome letter, whereby I find you abundant in courtesies of all natures... I have sent up two of your books which have much pleasured me... I shall much long to see you at this place, where you shall command the heart of your affectionate friend and servant	Edward Dering

4 Sir Edward Dering to Sir Robert Cotton, May 10, 1630. London, British Library,

This model facilitates answering a question like "How many letters did X send?" but does not facilitate answering "When did X write to Y?" To answer that question, the model must include information about time; an expanded model could consist of four attributes: addressee, content, sender, and date (Table 3).

Table 3. Expanded letter data model

Addressee	Content	Sender	Date
Sir	I received your very welcome letter,... where you shall command the heart of your affectionate friend and servant	Edward Dering	May 10 1630

Now the model contains some information about date and could answer the question "When did X write to Y?," but it could not answer the more specific question "What year did X write to Y?" because date has been treated as a single piece of information. Since a date consists of a month, a day, and a year, that structure can be added to the data model. Say one is also interested in the question "What closing phrases does X use when writing to Y?" In previous models, everything in the letter between the addressee's name and the sender's name has been treated as a single element called "content." Instead, the data model could highlight the "message + closing phrase" structure that is present in that content by separating the two. With these two changes, the new model is as follows:

Table 4. Final letter data model

Addressee	Message	Sender	Date			Closing phrase
			Month	*Day*	*Year*	
Sir	I received your very welcome letter,... where you shall command the heart of	Edward Dering	May	10	1630	your affectionate friend and servant

Not only does this structure work for this letter in particular; it could also be used to describe any number of similar letters. By identifying the general characteristics we are interested in asking questions about, rather than the specific content of a given letter, we are able to talk about and use pieces of information from the letter regardless of content.

Of course, not every letter follows this exact format. Some letters include less information—they may not be dated, or may not be signed, for example—and some letters may include more—a mailing address for either party,

models are best developed in the context of a larger sample of the data you are trying to model, not just a single instance.

Making a data model before making any kind of annotations helps ensure uniformity when it comes time to actually put data into the model. The decisions made in this step will determine what kinds of analyses are possible. A data model does not need to fully encode every aspect of the thing it is modelling, only the attributes one wishes to study. Conversely, it is important to remember that the information *not* chosen for inclusion in a model will *not* be available as an axis of analysis. Developing a data model involves thinking ahead and imagining what one wants to do with the data later, working two sides simultaneously: what structure does the data already have, and what questions or explorations does that suggest; and what structure does the data lack that I wish it had because of the questions or explorations I want to pursue?

One must also decide what to do with missing information. If one letter only has a year for the date, no month or day, how should that be handled in the data model? Are those fields left blank? Do they default to Jan 1? In the example letter, the addressee is referred to only as "Sir"—should the Addressee field be left as-is, or supplemented with information external to the letter to produce a name? How will viewers of the data know when such additions or alterations have been made? All such decisions should be documented so that conclusions drawn from the modelled data do not rely on (unintentionally) misleading information.

The process of data modelling also raises important questions about the nature of your data. For example, in the letter above, when the content was broken into a "message" + a "closing phrase," how was the delineation point determined? The present convention, the way we format "letters" over email, is that the closing phrase comes after the last full sentence of the message. In the example letter model, however, the split was not based on a sentence break, but on the *mise-en-page*—the words "your affectionate friend and servant" is the second half of a full sentence, but on the page it is set apart by whitespace. In a different letter that did not have the same formatting, how would the break point between message body and closing phrase be determined? Based on semantic similarity to phrases typically seen at the end of letters? In data modelling, there is rarely one right answer to these questions—that is why documenting the choices that are made during the data modelling phase is so crucial.

Data Modelling for Network Analysis

In the context of network analysis, data models always have two parts: nodes and edges. Continuing with the example of a letter, say we are developing a data model for a correspondence network. The information tracked is the same as in the previous model, but the way its structure is imagined is a little different. Since the two main "things" of interest in a correspondence network are people and letters, there are two categories of nodes, one called Person and one called Letter. Each Person can be connected to all the Letters in which they were participants. In order to retain information about the nature of each Person's role in each Letter, each edge can be labelled with either "Sender" or "Recipient."

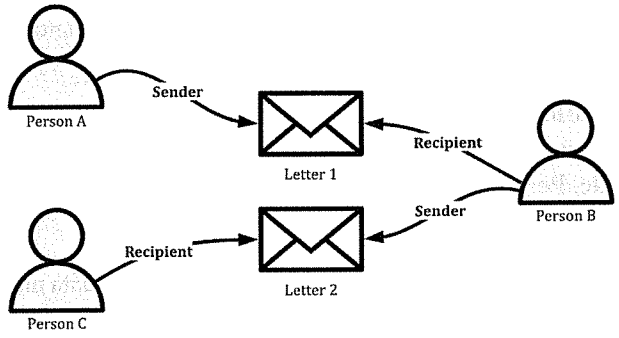


Figure 3. Sample bimodal correspondence network diagram, containing both people and letters as nodes. Diagram by author.

This is the basic form of a data model for network analysis: nodes each representing some "meaningful unit" of study, with a list of properties; edges representing the kinds of connections those units can have, with their own list of properties; and a set of rules about when edges connect nodes. To make sure the correct node for each edge can be determined, each Person and Letter also needs an identifier. Since there could be multiple people called "Edward Dering," the name is not a good option, and there is no immediately obvious unique identifier for a Letter, each node needs to be assigned a unique identifier. In tabular form, the network pictured in Figure 3 would look as follows:

Table 5. Sample correspondence network data model

NODES		
ID	*Label*	*Type*
A	"Person A"	Person
B	"Person B"	Person
C	"Person C"	Person
1	"Letter 1"	Letter
2	"Letter 2"	Letter

EDGES		
Source	*Target*	*Label*
A	1	Sender
B	1	Recipient
B	2	Sender
C	2	Recipient

This is a *bimodal, undirected* network—"bimodal" meaning there are two node types, and "undirected" meaning the edges do not have a direction. But this is not the only way to represent this same data as a network. If our primary interests are the relationships between *people*, we can collapse the Letter nodes into edges, each connecting two People nodes. To retain information about sender and recipient, which used to be encoded in the edges, we can give each Letter edge a direction, pointing from the sender to the recipient. Now instead of a bimodal, undirected network, we have a *unimodal, directed* one—unimodal because there is only one type of node, Person, and directed because the direction of the edges has meaning (Figure 4, Table 6.)

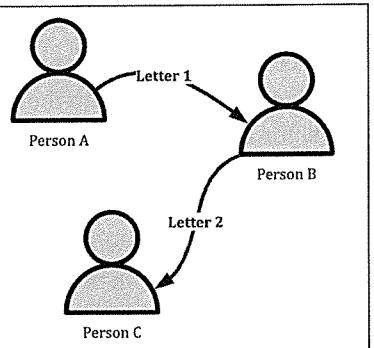


Figure 4. Sample unimodal correspondence network, containing only people as nodes. Diagram by author.

Table 6. Sample unimodal correspondence network data model

NODES		
ID	*Label*	*Type*
A	"Person A"	Person
B	"Person B"	Person
C	"Person C"	Person

EDGES		
Source	*Target*	*Label*
A	B	Letter 1
B	C	Letter 2

In later chapters, I refer to both bimodal and unimodal representations of each dataset. This example of a letter does not correspond directly to any of the datasets used for this project but is meant to provide an accessible introduction to the concepts discussed here. The specific data models for each dataset involved in this project are discussed in their respective chapters.

Uncertainty and Assumptions

As discussed in the introduction to this chapter, one of the most difficult parts of working with historical data in a digital context is dealing with the uncertainty inherent to the material. Records may be partial or hard to decipher; people may be referred to by only a first name or title; dates may be imprecise, covering a range of years, decades, or even centuries.

for fuzziness and leave open all the possibilities, but this strategy does not always translate well to a computational context. For example, the convention in manuscript studies of providing dates like "s. xii[1]" for the first half of the twelfth century or "s. xii[in]" for the "early" twelfth century is quite useful until you have to compare dates of many objects and place them on a timeline, raising questions like: what do "early" and "late" mean, numerically? does something dated to the "first half" of the century come before or after something dated to the "early" part of the century? and what about *circa* dates—should something dated to ca. 1300 be placed before, after, or co-located with something dated to precisely 1300?[5] For another example, when transcribing from manuscripts, scholars commonly use brackets to indicate expanded abbreviations and question marks to indicate indecipherable text. That is all well and good when reading a transcription, but when compiling names from a hundred transcriptions, how do we know whether "John," "J[ohn]," and "J[?]" are the same person? There are strategies developing for handling some uncertainty in network models,[6] but current tools are not up to the challenge. Instead of keeping fuzzy values, it may often be appropriate to make a more definitive determination (e.g., decide that "J[?]" is "John") add a "confidence" score or "uncertain" flag to your data model as a partial solution.

Working from secondary sources introduces an additional challenge, as one cannot always know how past "collaborators" have chosen to handle uncertainty themselves, or what their thresholds of confidence were—how certain did they need to be of something to list is as fact? What did they choose to include and exclude in their model? This is a problem any time one is building on the work of those that came before them, but it is an especially apparent one when working with data and is why documentation of decision-making processes is vital to data work: so those who may want to reuse it in the future are not left guessing.

Retrieving Machine-Readable Text

After developing a plan for the shape of the data, one must get text that can be read and manipulated by a computer to fill that shape. For a born-digital resource like the *DIMEV*, the text is already digital and ready to use. The

5 For further discussion of the digital manuscript dating problem, see Stokes, "The Problem of Digital Dating."

6 See, for example, Adar and Ré, "Managing Uncertainty"; Library of Congress,

810 PRYNNEANA—Pseudo Prynne.
Mr William Prynn | His Defence of | STAGE-PLAYS, | OR | A Retractation of a | former Book of his called | *Hiſtrio-Maſtix.* | [block of type-ornament] | [rule] | *London, printed in the Year* 1649. | [in border of type-ornament]

First edition. Quarto.

Collation: one sheet, signed A: title, p.[1]; blank, p.[2]; text, pp.3-8. (Pages numbered in parenthesis).

Bound with preceding number. 7⅝ x 5⅜ inches.

This *Retractation* is a palpable forgery which Prynne attempted to expose immediately by means of a broadside (reprinted Collier *Poetical Decameron* II, 321) headed: 'The vindication of William Prynne, Esquire from some scandalous papers and imputations newly printed and published to traduce and defame his reputation'†. As late as 2nd October 1649, the writer of *A serious epistle to Mr. William Prynne* pretended to think the *Retractation* was genuine (Hotson *Commonwealth and Restoration Stage* pp.41-2).

Only two other copies of this piece can be traced in the auction sales and it is doubtful if a half-dozen exist.

810 PRYNNEANA—Psevpo Prynne.
Mr William Prynn | His Defence of | STAGE-PLAYS, | OR | A Retractation of a | former Book of his called | Hiftrio-Maftix. | [block of type-ornament] | [rule] | London, printed in the Year 1649. | [in border of type-ornament]
FIRST EDITION. Quarto.
CoLzaTion: one sheet, signed A: title, p.[1]; blank, p.[2]; text, pp.3-8. (Pages numbered in parenthesis).
Bound with preceding number. 754 x 534 inches.
Tuts Retractation is a palpable forgery which Prynne attempted to expose immediately [...]

Figure 5. Example of Tesseract OCR results on an entry in the Pforzheimer catalogue, with errors highlighted. Image courtesy of Book Collection, Harry Ransom Center, The University of Texas at Austin.

editors of the *DIMEV* have already taken the print version—the *IMEV*—and converted it into a digital resource with machine-readable text, meaning I did not need to go through that process myself (instead, I had to find a way to download the *DIMEV* data—a process that will be discussed further). For physical, printed data sources, extra processing is required to convert the printed text into machine text.

If you have ever worked with PDF scans of physical documents and noticed that sometimes you can copy/paste the text from a PDF and sometimes you cannot, you have experienced the difference between machine-

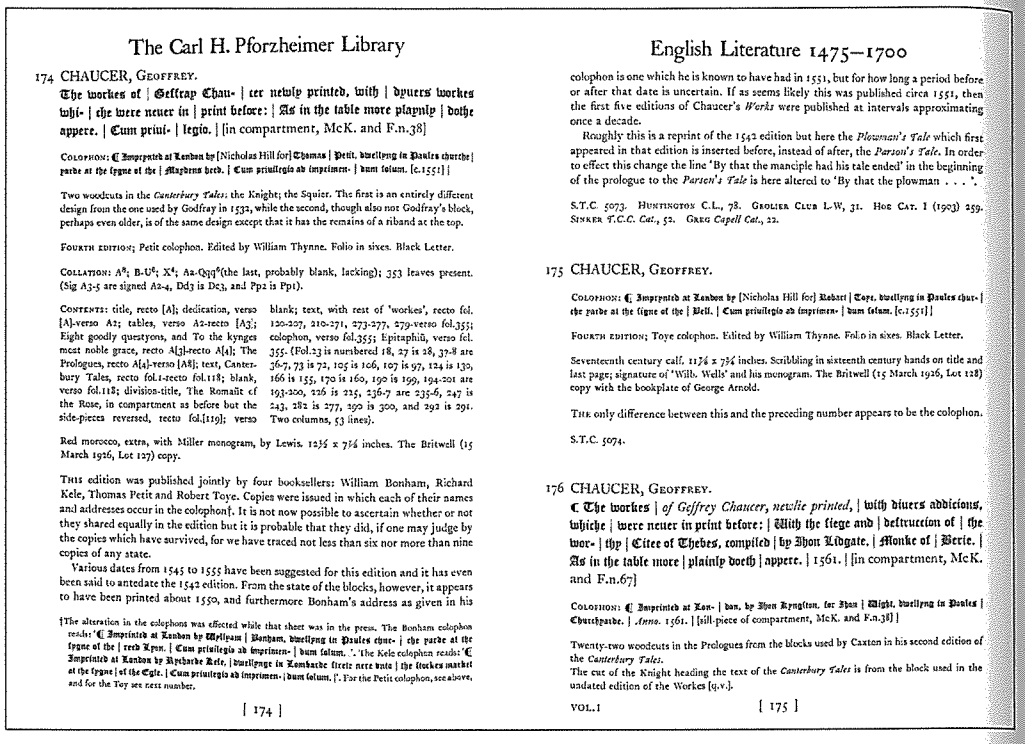
into an image file. To a human, the text is just as readable as it was on the physical page, but to a machine it is no different than a photograph of a landscape—just a collection of pixels. To turn the text from a scanned page into text that is fully readable, searchable, copyable, and manipulable on the computer, a technology called optical character recognition (OCR) is used.

OCR has been around for many years now, and there are many different tools available. Different OCR tools (often called OCR "engines") have been developed using different algorithms and training data, and are therefore better at reading certain kinds of text than others. For example, the average off-the-shelf OCR tool developed for use in business will not have a high degree of accuracy on old books printed in blackletter or manuscripts written in Caroline minuscule, but custom-developed OCR tools may be up to the task.[7] One of the most popular OCR engines is an open-source one called Tesseract. It is widely used because it is free, performs about as well as proprietary systems in terms of accuracy for simple documents, and can be used in many different programming languages.

Even for the materials like those I worked with in this project, which are printed books made in the last thirty years that use fairly standard typefaces, OCR is rarely 100% accurate. Scan quality, contrast, and the visual clarity of the original printed material all affect how accurate OCR will be. Even with high quality scans, quality can be difficult to predict. In the following example (Figure 5) taken from the Pforzheimer catalogue, Tesseract makes the same error beginning (human) students of book history make, mistaking the long-S for an "f." It also makes several less expected errors, like mistaking the "D" in "PSEUDO" for a "p" and rendering "Collation" as "CoLzaTion," perhaps indicating that it struggles with the small-capital font style.

Many OCR engines also struggle with layout—the more complicated the page layout and the more non-text elements present on the page, the more likely the OCR engine is to give inaccurate results. Multiple columns of text and hanging indents all present challenges. When presented with a two-up scan from the Pforzheimer catalogue (Figure 6), Tesseract successfully recognizes the divide between the two pages, but it also detects the hanging indents as belonging to their own columns and processes it as if it were a 4-column document, beginning: "174 / 175 / 176 / The Carl H. Pforzheimer Library / CHAUCER, Geoffrey..." In the body of entry 174, Tesseract has the opposite problem of detecting too *few* columns when it reaches the "Con-

7 On OCR of blackletter, see Ströbel and Clematide, "Improving OCR of Black Letter." On OCR of Caroline minuscule, see Hawk, Karaisl, and White, "Modelling Medieval



The Carl H. Pforzheimer Library

English Literature 1475–1700

174 CHAUCER, Geoffrey.

The workes of | Geffray Chau- | cer newly printed, with | diuers workes whi- | che were neuer in | print before: | As in the table more playnly | dothe appere. | Cum priui- | legio. | [in compartment, McK. and F.n.38]

Colophon: € Imprynted at London by [Nicholas Hill for] Thomas | Petit, dwellyng in Paules churche | yarde at the sygne of the | Maydens hed. | Cum priuilegio ad imprimen- | dum solum. [c.1551] |

Two woodcuts in the *Canterbury Tales*: the Knight; the Squier. The first is an entirely different design from the one used by Godfray in 1532, while the second, though also not Godfray's block, perhaps even older, is of the same design except that it has the remains of a riband at the top.

Fourth edition; Petit colophon. Edited by William Thynne. Folio in sixes. Black Letter.

Collation: A^8; B-U^6; X^4; Aa-Qqq^6(the last, probably blank, lacking); 353 leaves present. (Sig A3-5 are signed A2-4, Dd3 is Dc3, and Pp2 is Pp1).

Contents: title, recto [A]; dedication, verso [A]-verso A2; tables, verso A2-recto [A3]; Eight goodly questyons, and To the kynges most noble grace, recto A[3]-recto A[4]; The Prologues, recto A[4]-verso [A8]; text, Canterbury Tales, recto fol.1-recto fol.118; blank, verso fol.118; division-title, The Romaūt of the Rose, in compartment as before but the side-pieces reversed, recto fol.[119]; verso blank; text, with rest of 'workes', recto fol. 120-207, 210-271, 273-277, 279-verso fol.355; colophon, verso fol.355; Epitaphiū, verso fol. 355. (Fol.23 is numbered 18, 27 is 28, 37-8 are 36-7, 73 is 72, 105 is 106, 107 is 97, 124 is 130, 166 is 155, 170 is 160, 190 is 199, 194-201 are 193-200, 226 is 225, 236-7 are 235-6, 247 is 243, 282 is 277, 290 is 300, and 292 is 291. Two columns, 53 lines).

Red morocco, extra, with Miller monogram, by Lewis. 12⅜ x 7¾ inches. The Britwell (15 March 1926, Lot 127) copy.

This edition was published jointly by four booksellers: William Bonham, Richard Kele, Thomas Petit and Robert Toye. Copies were issued in which each of their names and addresses occur in the colophon†. It is not now possible to ascertain whether or not they shared equally in the edition but it is probable that they did, if one may judge by the copies which have survived, for we have traced not less than six nor more than nine copies of any state.

Various dates from 1545 to 1555 have been suggested for this edition and it has even been said to antedate the 1542 edition. From the state of the blocks, however, it appears to have been printed about 1550, and furthermore Bonham's address as given in his colophon is one which he is known to have had in 1551, but for how long a period before or after that date is uncertain. If as seems likely this was published circa 1551, then the first five editions of Chaucer's *Works* were published at intervals approximating once a decade.

Roughly this is a reprint of the 1542 edition but here the *Plowman's Tale* which first appeared in that edition is inserted before, instead of after, the *Parson's Tale*. In order to effect this change the line 'By that the manciple had his tale ended' in the beginning of the prologue to the *Parson's Tale* is here altered to 'By that the plowman . . . '.

S.T.C 5073. Huntington C.L., 78. Grolier Club L-W, 31. Hoe Cat. I (1903) 259. Sinker *T.C.C. Cat.*, 52. Greg *Capell Cat.*, 22.

†The alteration in the colophon was effected while that sheet was in the press. The Bonham colophon reads: '€ Imprinted at London by Wellyam | Bonham, dwellyng in Paules chur- | che yarde at the sygne of the | rede Lyon. | Cum priuilegio ad imprimen- | dum solum. '. The Kele colophon reads: '€ Imprinted at London by Rychardе Kele, | dwellynge in Lombarde strete nere vnto | the stockes market at the sygne | of the Egle. | Cum priuilegio ad imprimen- | dum solum. '. For the Petit colophon, see above, and for the Toy see next number.

[174]

175 CHAUCER, Geoffrey.

Colophon: € Imprynted at London by [Nicholas Hill for] Robart | Toye, dwellyng in Paules chur- | che yarde at the signe of the | Bell. | Cum priuilegio ad imprimen- | dum solum. [c.1551] |

Fourth edition; Toye colophon. Edited by William Thynne. Folio in sixes. Black Letter.

Seventeenth century calf. 11¾ x 7¾ inches. Scribbling in sixteenth century hands on title and last page; signature of 'Wilb. Wells' and his monogram. The Britwell (15 March 1926, Lot 128) copy with the bookplate of George Arnold.

The only difference between this and the preceding number appears to be the colophon.

S.T.C. 5074.

176 CHAUCER, Geoffrey.

€ The workes | *of Geffrey Chaucer, newlie printed,* | **with diuers addicions, whiche | were neuer in print before: | With the siege and | destruccion of | the wor- | thy | Cитee of Thebes, compiled | by Jhon Lydgate. | Monke of | Bery. | As in the table more | plainly doeth | appere.** | 1561. | [in compartment, McK. and F.n.67]

Colophon: € Imprinted at Lon- | don, by Jhon Kyngston, for Jhan | Wight, dwellyng in Poules | Churchyarde. | *Anno.* 1561. | [sill-piece of compartment, McK. and F.n.38] |

Twenty-two woodcuts in the Prelogues from the blocks used by Caxton in his second edition of the *Canterbury Tales*.

The cut of the Knight heading the text of the *Canterbury Tales* is from the block used in the undated edition of the Workes [q.v.].

VOL. I

[175]

Figure 6. Example of two-up scan from Pforzheimer catalogue. The multi-column layout with hanging indentations presents challenges to automatic text recognition. Image courtesy of Book Collection, Harry Ransom Center, The University of Texas at Austin.

tents" section. Rather than reading that section as two small columns, it reads straight across, producing: "Contents: title, recto [A]; dedication, verso blank; text with..."

OCR results can be difficult to predict, and impossible to guarantee when working with a large number of images with complicated typography or varying scan quality—the settings that work for some images may not work for others. While these results are not perfect, they are good enough that correcting the parts of this document needed for further processing will still result in a huge amount of time saved as compared with transcribing the text manually.[8]

8 Some have even argued the merits of working with fully uncorrected OCR—see

Tite_entries_entry-1175.0_titus-fviii	7/21/2022 2:18 PM	Text Document	1 KB
Tite_entries_entry-1176.0_titus-fix	7/21/2022 2:18 PM	Text Document	1 KB
Tite_entries_entry-1177.0_titus-fx	7/21/2022 2:18 PM	Text Document	1 KB
Tite_entries_entry-1179.0_titus-fxi	7/21/2022 2:18 PM	Text Document	1 KB
Tite_entries_entry-1180.0_titus-fxii	7/21/2022 2:18 PM	Text Document	1 KB
Tite_entries_entry-1181.0_titus-fxiii	7/21/2022 2:18 PM	Text Document	1 KB
Tite_entries_entry-1183.0_titus-fxiv	7/21/2022 2:18 PM	Text Document	2 KB
Tite_entries_entry-1184.0_domitian-i	1/28/2023 9:48 AM	Text Document	2 KB
Tite_entries_entry-1185.0_domitian-ii	7/21/2022 2:18 PM	Text Document	3 KB
Tite_entries_entry-1188.0_domitian-iii	7/21/2022 2:18 PM	Text Document	1 KB

Figure 7. Selection of Cotton loan records after segmentation, split into individual text files.

```
?* DOMITIAN    I  Isidore,  De  naturis rerum; Gerald  of
        Wales, Itinerarium et descriptio Cambriae; etc.
      Catalogues:  6018, no. 412; 36789,  fols 90-91; 36682
        (although the 1656/7 checklist on fol. 2 records '[n]Joe
        contents in the Catalogue: noe booke in the place' under
        Domitian).
      Previous ownership:  (of fols 2-55, artt. 1-6) John Dee
        (Roberts  & Watson,  Dee, M94); (of fols 56-160, artt.
        7-13) John  Prise, ? Gregory Prise, Dee (Ker, 'Prise', 3
        and  P.10; Roberts & Watson, Dee, DM26, p. 17; and see
        the entry for Dom. V).
      Loans: William  Camden,  1606 (10.11: this, Vesp. A.II or
        B.XXV.  Cf. Jul. C.IH, fol. 65); John Selden, 1638 (155.3:
        this, Jul. B.XIII, Tib. B.XIII, C.III, Nero D.VIII, Vit.
        C.X, EV, VII, Dom. V, Cleo. D.V or Faust. C.IV).
      Extracts, marked as from this manuscript, are in James
        Ware's notebook, Add. MS  4783, fol. 56.
      Annexes  3a, 5a, 7b.
```

Figure 8. Example of a single Cotton loan record after segmentation. The segmentation process results in a plain text file for each book's entry.

Segmenting

During the data modelling step, the "meaningful units" of data were identified—what can be thought of as discrete "things" in the dataset, or the level at which metadata is gathered. In the *DIMEV*, for example, the units are Texts and Books; in the Cotton and Pforzheimer records, the units are Books. But even after doing OCR, these units are not separated from one another; the data is still just a single giant blob, which is difficult to work with. If one were to proceed to annotation with the raw OCRed text, the entire text would have to be annotated at once and the divisions between the units of data manually marked. To ease annotation by making the amount of data that needs to be handled at any given time small, and to facilitate collaboration if you had a team of people annotating, it is helpful to first segment the data set into its meaningful units.

Of the case studies that comprise this project, only the two born-print resources require this step. The *DIMEV* is already segmented into Books and Texts—each has its own page on the *DIMEV* website with its individual metadata, and each Text and Book already has a unique identifier. For the print-born resources, this is a much more complicated step. Each of these data sets is large, hundreds of pages. To facilitate annotation, it was necessary to segment the images by unit of text—single book's entry in the Pforzheimer catalogue or edition of Cotton's loan records—separating each into its own

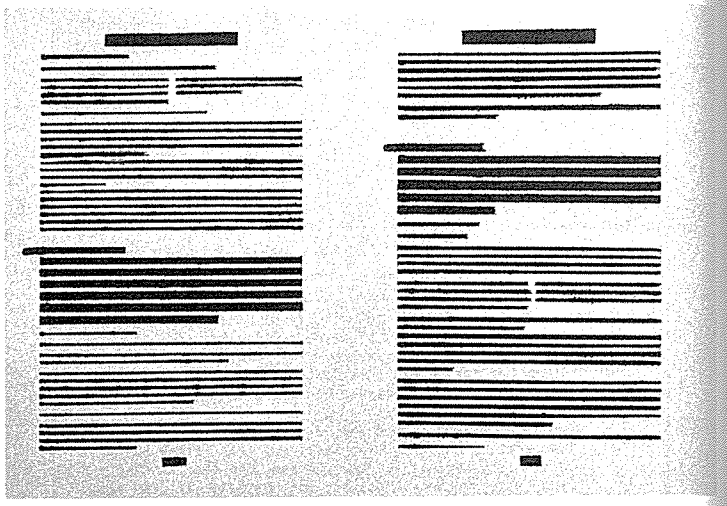


Figure 9. A painted abstraction of the layout of two pages from the Pforzheimer catalogue. Even with no text, the painting is recognizable as pages of text, major features of the page layout are identifiable, and reasonable guesses about conceptual grouping can be made. Image by author.

Automating the segmentation process is no small task. Understanding the meaning of text on a page is one of those tasks that humans are very good at but with which computers struggle. It is also a separate task from simply recognizing the text, as OCR does. When humans look at the page, we rely on cues and our knowledge of print publication conventions to understand the layout. Just as our internalized understanding of the textual conventions of a letter make it easy to recognize one, our internalized understanding of paratextual conventions and pattern recognition skills make it easy to derive meaning from the layout of a page.[9]

For example, in the digitized Pforzheimer catalogue (Figure 6), it is fairly simple for a human to recognize that the numbered hanging indents indicate the starts of new sections, and that everything between those numbered indents belongs together. Even without reading the words on the page, people who have spent time around modern books can identify that this image shows two separate pages, that the larger words at the top of the page are a running header, that the small numbers centred at the bottom are page numbers, that the larger text at the start of each section probably serves as some kind of heading or introduction, and that the smaller columns within entries should probably be read consecutively rather than together.

This task, often called layout analysis or document analysis, is much harder for machines. In order to be able to "understand" the page layout of a scan, a computer needs to be taught the rules of page layout. Such work is being undertaken by researchers in a variety of fields, including business, economics, and medicine, as digitizing large volumes of forms and paper-

work requires the ability to parse those forms automatically. In DH, many researchers who make use of newspapers are working on layout analysis for article segmentation.[10] Form extraction is essential for human rights and justice work, as well, where processing large volumes of paper records may be necessary to uncover systemic abuses.[11]

Unlike simple OCR, for which there are many well-established tools, there is not a "canonical" tool in the DH tool belt for segmentation. Many OCR tools, including Tesseract, come with a moderate level of layout detection capabilities (see Figure 10), but those systems are designed mostly to help with the ordering of text, not to produce the most conceptually meaningful divisions of the page. Expensive OCR tools like Abbyy FineReader may perform better out-of-the-box in this respect, but their cost is prohibitive for many users, especially those undertaking small projects without access to funding.

A few tools exist that use artificial intelligence that can be *trained* to recognize certain kinds of documents. Transkribus—a tool which primarily markets itself as handwriting OCR software—has such machine learning-driven layout analysis capabilities, but the learning curve for Transkribus is steep. As with any machine learning tool, training the model to understand a particular use case requires a substantial amount of up-front manual annotation from which the algorithm can learn, and even then, the layout analysis feature primarily supports altering the reading order of the OCR rather than identifying intellectually meaningful units of text. Other machine learning-driven solutions are in active development—since my project began, a new suite of tools called Layout Parser has released with a "zoo" of pre-trained models to choose from.[12] While I have not had the opportunity to test Layout Parser on my own data, it appears from demos that the default models also focus on finding paragraph and column divisions to define reading order, but it may be possible to achieve the desired segmentation through a custom-trained model using the Layout Parser framework.

Though a machine learning-based approach may seem promising, and with time such tools will only get better and easier to use "straight out-of-the-box," right now they invariably require custom training to work. Training a machine learning algorithm takes a great deal of time, labour, and

10 See, for example, Barman et al., "Combining Visual and Textual Features"; Naoum, Nothman, and Curran, "Article Segmentation in Digitised Newspapers."

11 For more examples of the applications and challenges of document analysis, see Stray, "If You Want to Use AI/ML for Good"

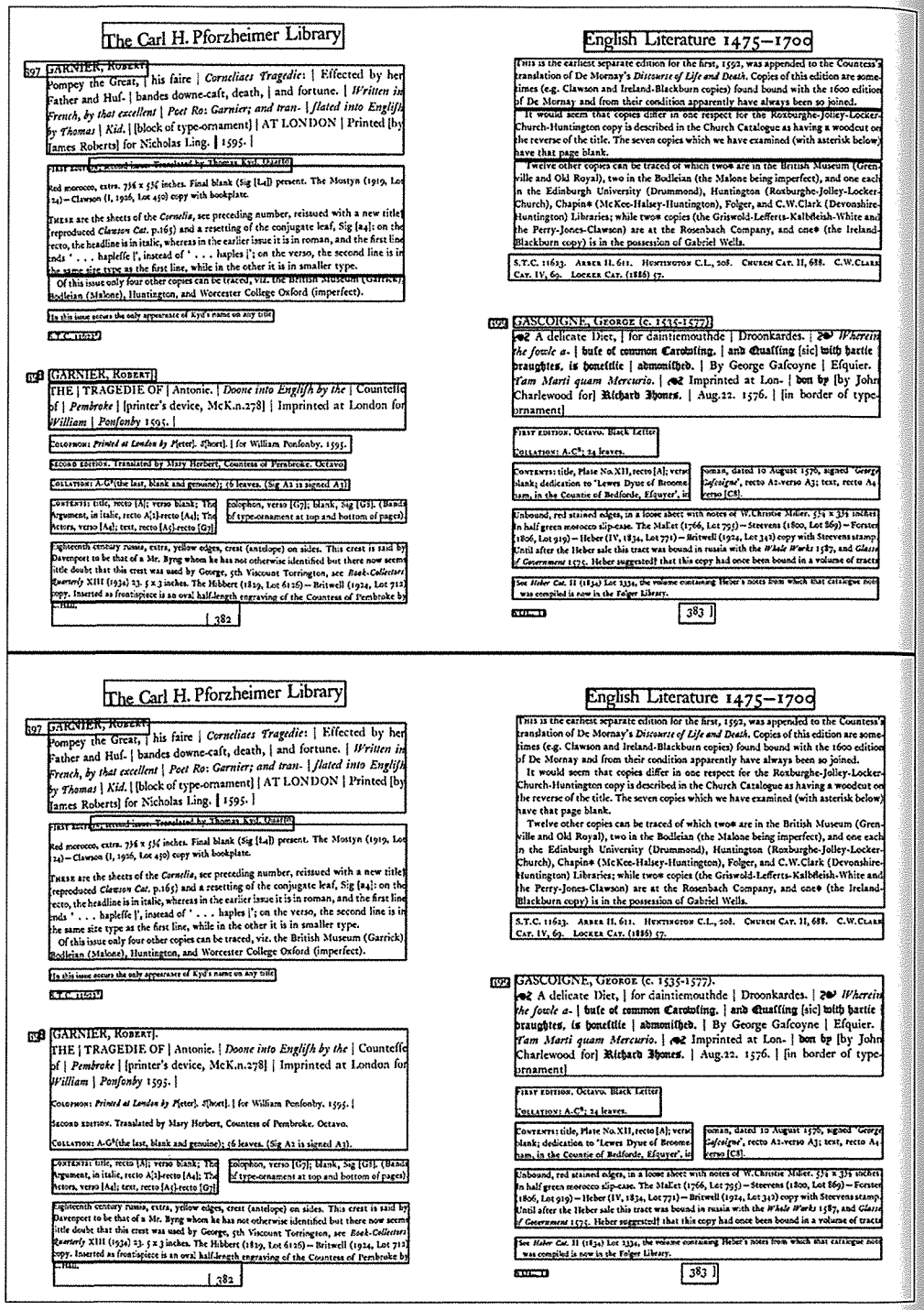


Figure 10. Visualized results from two different Tesseract layout analysis strategies, top and bottom. Neither strategy

training that not all (or even many) projects can afford—I certainly could not make use of such tools within the constraints of this project, even with substantial experience with machine learning. With no solution meeting my needs, I undertook development of a custom tool.[13]

Developing a New Segmentation Tool

When I first discussed the development of a segmentation tool on Twitter, several colleagues assumed I would be taking a machine learning approach, like Transkribus and Layout Parser do. I understand where they were coming from. Machine learning and artificial intelligence are hot topics right now, in digital humanities and in general, and machine learning projects are attractive to funding agencies. But machine learning is simply not the best solution for every problem.

Minimal computing is a movement in DH that asks us to resist the urge to over-engineer and to prioritize "ease of use, ease of creation, increased access, and reductions in computing" when solving technical problems.[14]

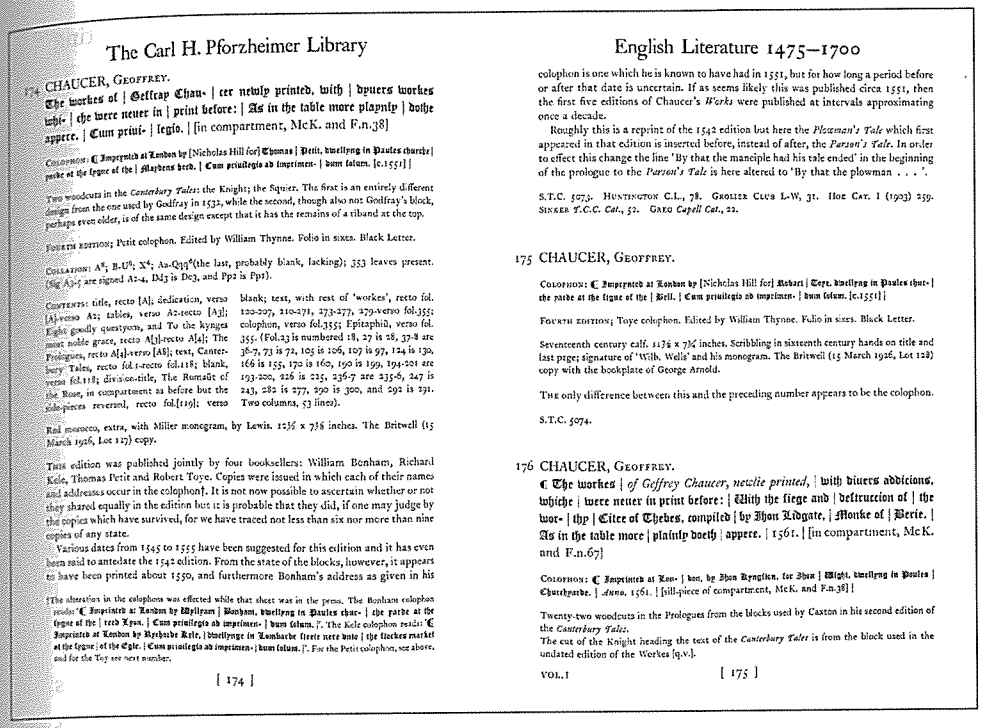
1. **Ease of use.** A machine learning-based solution to segmentation could in theory be made easy to use, in that it would not require the end users to lay out any rules for segmentation but would instead "magically" produce results from an input based on its learning. In practice, extant machine learning-based tools for segmentation (such as Layout Parser) require significant familiarity with Python, reading technical documentation, and the processes of machine learning. Unless the user's data closely matches the kind of data used for training, some retraining or fine-tuning is likely to be required. So, while theoretically easy to use, in practice machine learning can make a tool harder to use effectively.
2. **Ease of creation.** Machine learning solutions are costly to create. Effective machine learning requires quality training data, and lots of it. For most problems, document segmentation included, labelled training data is not simply lying around—it must be created, manually, by humans. The process of labelling data, training a model, testing and refining it, and making it usable for others is time-intensive and labori-

13 Although it is usually best to design workflows with as little custom software as possible, this is a gap that could not be filled with something else. Since beginning work on a segmentation tool, I have spoken or worked with many people developing projects which would also benefit from this tool: one using medieval cathedral records, one using legal documents, one using newspapers, one using a library catalogue, and one using copyright office records. The application potential is broad.

ous. When developing a tool driven by machine learning, one needs to provide either a *great* starting model that is trained on a wide variety of datasets, which takes a lot of development time and resources, or a system for the user to be able to train their own model, which takes lots of *their* time and resources.

3. **Increased access.** The same problems that can make machine learning tools difficult to use and difficult to create contribute to reduced access. If a tool requires significant human resources to label data for training, or to correct the outputs from a model not suited to the data on which it is being used, that tool is not accessible to users without a team or budget. If a tool requires significant programming capability to use, it is not accessible to those without that knowledge or access to team members who do.
4. **Reductions in computing.** Deep learning, the variety of machine learning that is popular right now and backs tools like Layout Parser, is computationally intensive. While there are other machine learning methods that are less so, and certainly the scale of a digital humanities project's use would be small potatoes compared to mindbogglingly consumptive and environmentally costly large language models (e.g., ChatGPT), using machine learning for problems that could be solved via less computationally intensive methods would *not* be a reduction in computing.

For all these reasons, developing "dumb" tools—ones that do not learn as you use them but that are fast to create and perform well at a narrow set of tasks—is often the right answer. Put another way, minimal computing is "a heuristic comprising four questions to determine what is, in fact, necessary and sufficient when developing a digital humanities project under constraint: 1) 'what do we need?'; 2) 'what do we have'; 3) 'what must we prioritize?'; and 4) 'what are we willing to give up?'"[15] In this case, 1) I needed a way to segment my documents; 2) I had a relatively small dataset and am a team of one; 3) I needed to prioritize getting my data processed so I could move on to using it; and 4) I was willing to give up some amount of time to generalize the toolkit for my future projects (and, hopefully, others'), but not willing to give up the amount of time required to develop robust training datasets. The results: machine learning was out; an old-fashioned rules-based tool was in.



The Carl H. Pforzheimer Library

English Literature 1475–1700

174 CHAUCER, Geoffrey.

The workes of | Geffrey Chau- | cer newly printed, with | dyuers workes whi- | che were neuer in | print before: | As in the table more playnly | doth appere. | Cum priui- | legio. | [in compartment, McK. and F.n.38]

Colophon: ¶ Imprynted at London by [Nicholas Hill for] Thomas | Petit, dwellyng in Paules churche | yarde at the ſygne of the | Maydens hede. | Cum priuilegio ad imprimen- | dum ſolum. [c.1551] |

Two woodcuts in the *Canterbury Tales*: the Knight; the Squier. The first is an entirely different design from the one used by Godfray in 1532, while the second, though also not Godfray's block, perhaps even older, is of the same design except that it has the remains of a riband at the top.

Fourth edition; Petit colophon. Edited by William Thynne. Folio in sixes. Black Letter.

Collation: A⁸; B-U⁶; X⁴; Aa-Qq10⁸(the last, probably blank, lacking); 353 leaves present. (Sig A3-5 are signed A2-4, DA3 is De3, and Pp2 is Pp3).

Contents: title, recto [A]; dedication, verso [A]-verso A2; tables, verso A2-recto [A3]; Eight goodly questyons, and To the kynges most noble grace, recto A[3]-recto A[4]; The Prologues, recto A[4]-verso [A8]; text, Canterbury Tales, recto fol.1-recto fol.128; blank, verso fol.128; division-title, The Rumaũt of the Rose, in compartment as before but the side-pieces reversed, recto fol.[129]; verso blank; text, with rest of 'workes', recto fol. 130-207, 210-271, 273-277, 279-verso fol.355; colophon, verso fol.355; Epitaphii, verso fol. 356. (Fol.236 is numbered 18, 27 is 28, 37-8 are 36-7, 73 is 72, 105 is 106, 107 is 97, 124 is 130, 166 is 155, 170 is 160, 190 is 199, 194-201 are 193-200, 226 is 225, 236-7 are 235-6, 247 is 243, 282 is 277, 290 is 300, and 292 is 291. Two columns, 53 lines).

Red morocco, extra, with Miller monogram, by Lewis. 12⅜ x 7⅜ inches. The Britwell (15 March 1926, Lot 127) copy.

This edition was published jointly by four booksellers: William Bonham, Richard Kele, Thomas Petit and Robert Toye. Copies were issued in which each of their names and addresses occur in the colophon†. It is not now possible to ascertain whether or not they shared equally in the edition but it is probable that they did, if one may judge by the copies which have survived, for we have traced not less than six nor more than nine copies of any state.

Various dates from 1545 to 1555 have been suggested for this edition and it has even been said to antedate the 1542 edition. From the state of the blocks, however, it appears to have been printed about 1550, and furthermore Bonham's address as given in his

†The alteration in the colophons was effected while that sheet was in the press. The Bonham colophon reads: '¶ Imprinted at London by Whyliam | Bonham, dwellyng in Paules chur- | che parde at the ſygne of the | rede Lyon. | Cum priuilegio ad imprimen- | dum ſolum.'. The Kele colophon reads: '¶ Imprinted at London by Raybalde Kele, | dwellynge in Lombarde ſtreete nere vnto | the ſtockes market at the ſygne | of the Egle. | Cum priuilegio ad imprimen- | dum ſolum.'. For the Petit colophon, see above, and for the Toy see next number.

[174]

colophon is one which he is known to have had in 1551, but for how long a period before or after that date is uncertain. If as seems likely this was published circa 1551, then the first five editions of Chaucer's *Works* were published at intervals approximating once a decade.

Roughly this is a reprint of the 1542 edition but here the *Plowman's Tale* which first appeared in that edition is inserted before, instead of after, the *Parson's Tale*. In order to effect this change the line 'By that the manciple had his tale ended' in the beginning of the prologue to the *Parson's Tale* is here altered to 'By that the plowman . . . '.

S.T.C. 5073. Huntington C.L., 78. Grolier Club L.W, 31. Hoe Cat. I (1903) 259. Singer *T.C.C. Cat.*, 52. Grey *Capell Cat.*, 22.

175 CHAUCER, Geoffrey.

Colophon: ¶ Imprynted at London by [Nicholas Hill for] Robart | Toye, dwellyng in Paules chur- | che parde at the ſigne of the | Bell. | Cum priuilegio ad imprimen- | dum ſolum. [c.1551] |

Fourth edition; Toye colophon. Edited by William Thynne. Folio in sixes. Black Letter.

Seventeenth century calf. 11⅞ x 7¼ inches. Scribbling in sixteenth century hands on title and last page; signature of 'Wilh. Wells' and his monogram. The Britwell (15 March 1926, Lot 128) copy with the bookplate of George Arnold.

The only difference between this and the preceding number appears to be the colophon.

S.T.C. 5074.

176 CHAUCER, Geoffrey.

¶ The workes | *of Geffrey Chaucer, newlie printed,* | with diuers additions, whiche | were neuer in print before: | With the ſiege and | deſtruccion of | the wor- | thy | Cite of Thebes, compiled | by Iohn Lidgate, | Monke of | Berie. | As in the table more | plainly doeth | appere. | 1561. | [in compartment, McK. and F.n.67]

Colophon: ¶ Imprinted at Lon- | don, by Iohn Kyngſton, for Iohn | Wight, dwellyng in Poules | Churchyarde. | *Anno.* 1561. | [fill-piece of compartment, McK. and F.n.38] |

Twenty-two woodcuts in the Prologues from the blocks used by Caxton in his second edition of the *Canterbury Tales*.

The cut of the Knight heading the text of the *Canterbury Tales* is from the block used in the undated edition of the Workes [q.v.].

VOL. I

[175]

Figure 11. Example of two-up scan from Pforzheimer catalogue. Note the way whitespace in the page layout conveys structural information. Image courtesy of Book Collection, Harry Ransom Center, The University of Texas at Austin.

Developing my own segmentation tool required me to consider how *I* understand page layout—how meaning in printed books is communicated visually, not just textually. After gathering examples of many different types of books and documents, I observed that—ignoring the meaning of the words—only a small handful of features are key to understanding layout: the whitespace between lines, indentation on the margin, and patterns of text around those indentations and gaps.

Consider, in the example pages from the Pforzheimer catalogue (Figure 11), the way whitespace guides interpretation of the page. The largest gap between text is down the middle, dividing two pages. The next largest gaps precede lines which intrude into the left margin, marking the start of a new catalogue entry. The next smallest gaps are between paragraphs or sub-columns of text, the next smallest are between lines of text, and the very smallest are gaps between individual characters. Some of these gaps run vertically, and some run horizontally.

Given these observations, a small set of rules and a procedure for separating catalogue entries in this image can be devised:

1. Split at the largest vertical-running gap to get two separate pages.
2. On each page, ignore the text above the first small gap (the running header), and below the last small gap (the page numbers).
3. When a medium-size horizontal gap is seen, a new entry starts with the following text. When the next similar gap is seen, moving down the page, that entry has ended, and a new one begins.
4. At the top of a page (after ignoring the header), there are two possibilities; either we see:
    a. A line that juts into the left margin, in which case another new entry has started.
    b. A line that is aligned with the margin, in which case the text that follows is part of an entry that began on an earlier page.

In a world of perfect OCR, the only rule that would be necessary is one describing the beginning of the text for each entry—a number and some capital letters (the catalogue number and author's last name). But OCR is rarely completely accurate, and there is no guarantee we will not see that same pattern of characters elsewhere. Some printed documents do not have consistent textual patterns marking the beginning of a new section but instead use whitespace combined with other patterns (like bolding, italics, or underlining) that OCR is unable to account for. Therefore, just as printers include whitespace to aid our eyes in parsing the page, whitespace must be used to aid the computer's "eyes."

A machine learning-based approach to this task would be to have the user draw boxes around the chunks of text they care about, and then slowly learn from their annotations how much whitespace is meaningful. Instead of giving the computer examples and letting it learn the rules through training, the tool I developed allows users to write these rules directly, segment and OCR a set of images according to the rules, and make corrections to both the segmentation and OCR simultaneously.

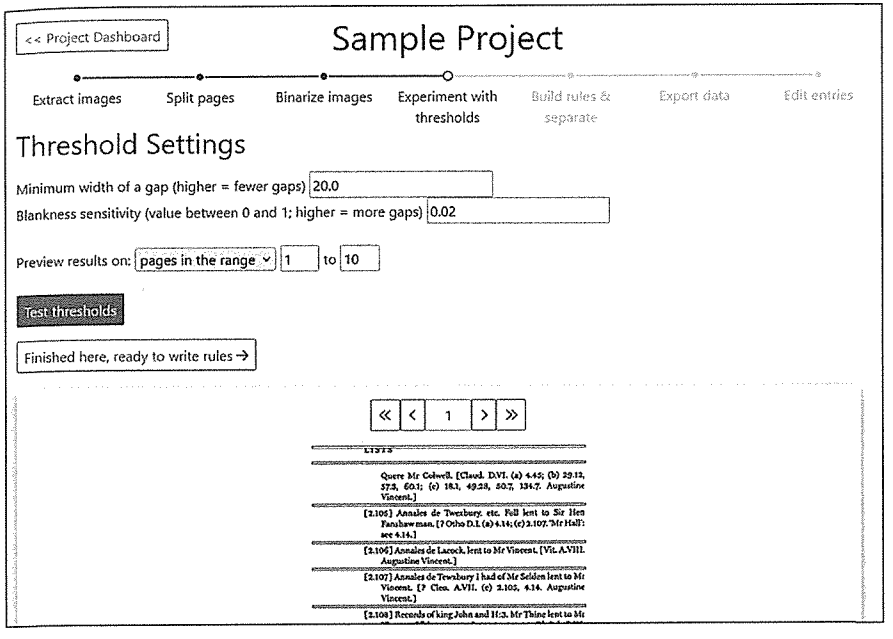


Figure 12. The user interface for the "experiment with thresholds" step of document segmentation. Image by author.

Workflow

The document segmentation application I developed runs in a web browser and takes the user through the following workflow:[16]

1. **Extract images**: A PDF is converted into a series of JPG images.
2. **Split pages**: If there are two columns of text on a page, as is the case with the above Pforzheimer example, the user indicates where on the image the column divide is, and the software splits each image into two separate images.
3. **Binarize images**: Images are converted from colour or greyscale to binary matrices (i.e., each pixel becomes fully black or fully white.)
4. **Experiment with thresholds**: The user adjusts two parameters until gap detection returns the desired results: the width of a gap in pixels, and the "blankness sensitivity"—the percentage of pixels in a horizontal slice of the image that can be black for that space to still be considered "blank." (Figure 12)

16 The tool is available for download at https://github.com/lizfischer/document-

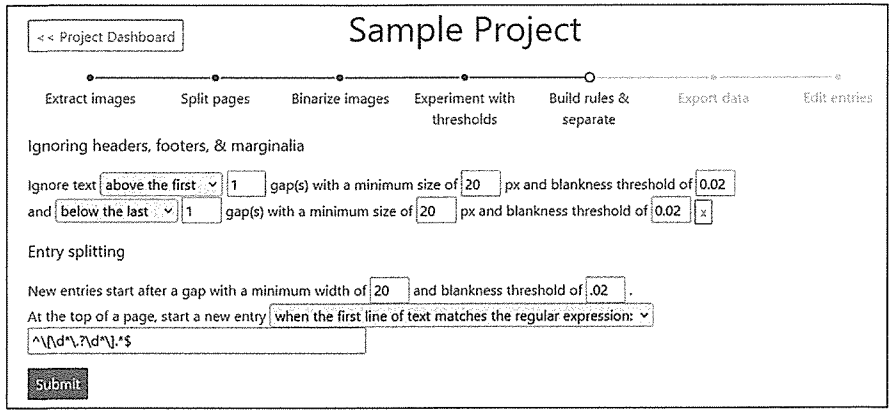


Figure 13. The user interface for the "build rules" step of document segmentation. Image by author.

5. **Build rules & separate**: Using the thresholds they settled on in the previous step, the user builds three rules for segmentation (Figure 13):
    a. Which (if any) text should be ignored at the top or bottom of each page,
    b. Which gaps indicate the start of a new entry, and
    c. What happens at the start of a new page—always start a new entry, never start a new entry, or start a new entry if the text matches a specified pattern.

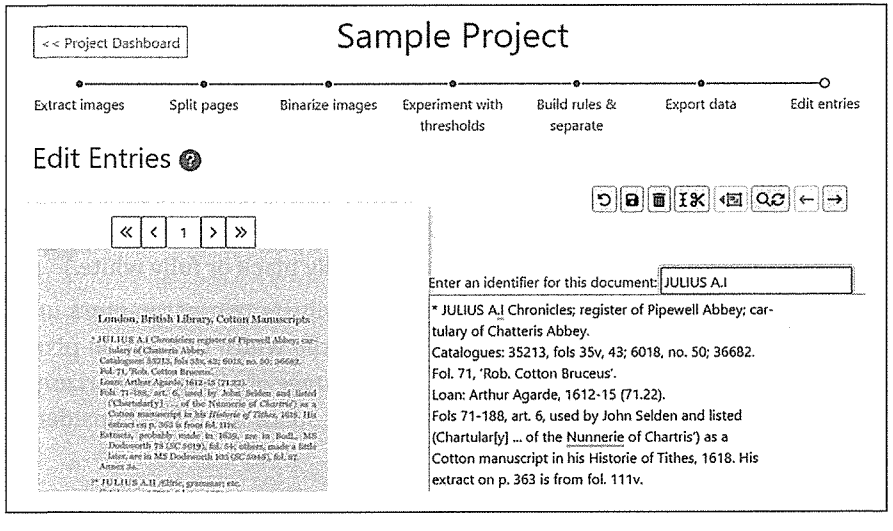


Figure 14. The user interface for the "edit entries" step

Once these rules are set, the software OCRs each image and divides the OCRed text based on the segmentation rules.

6. **Export data**: The user downloads a zipped folder containing one text file for each segment of their document.
7. **Edit entries (optional)**: The user may review OCR and segmentation results side by side with the original image and make manual corrections to the OCR text or the segmentation splits (by joining, splitting, or deleting segments). The user may also specify a custom identifier for each segment. (Figure 14)

The result of this workflow is a set of text files, each containing the full text of a single segment, ready to be further processed or annotated individually.

Quality of Results

Simple ground-truth testing over inputs from four different source documents indicates this whitespace-based method of segmentation performs on average 57% better than textual pattern recognition (through regular expressions, or regex) alone. The quality of both methods was measured in two ways: precision and recall. Precision measures how good a system is at returning *only* correct results (i.e., minimizing false positives), while recall measures how good a system is at returning *all* the results you wanted (i.e., maximizing true positives). A third score, called F1, is the harmonic mean of precision and recall, a balance of the two.

Table 7. Quality of segmentation results using regular expressions only and whitespace-based segmentation

	Regex Only	Whitespace Segmentation
Precision	45.12%	91.67%
Recall	75.51%	86.27%
F1	56.49%	88.89%

Using only regular expressions to segment based on patterns in OCR text had a precision of around 45%, meaning the resulting segmentation divisions were correct divisions 45% of the time; regex alone had a recall of around 76%, meaning that it found roughly three-quarters of the total divisions expected. My whitespace-based segmentation tool had a precision of around 92%, meaning the divisions it produced were correct the vast majority of the time; my tool had a recall of around 86%, meaning it found most of the divisions I hoped it would find. That brings the F1 score of the regex-

only approach to just over 56%, and the F1 score of my tool to just under 89%—a 57% improvement in accuracy.

Annotating

In this workflow, annotation enhances data by adding structure and metadata to the OCRed text. Annotation begins with scanned, segmented documents and results in marked-up documents, each containing the key pieces of information necessary for our analysis labelled. Like segmentation, annotation is about encoding information about the source material that is evident to humans but difficult for machines to see. To the computer, a blob of text is a blob of text unless we provide instructions for interpreting that text. Annotation moves a document from being "just text" to being analyzable data. In the context of digital humanities, the term "markup" is often used for a kind of annotation common to the field. Digital editions make extensive use of markup—usually in the form of XML standards like TEI—to encode differences in texts. Computational Linguistics makes extensive use of markup through Part of Speech Tagging and similar kinds of semantic annotation.

Manual annotation is a time-consuming process. Theoretically, one could handle annotation automatically by writing rules (like we did for segmentation) about what pieces of the text mean. Using regular expressions,

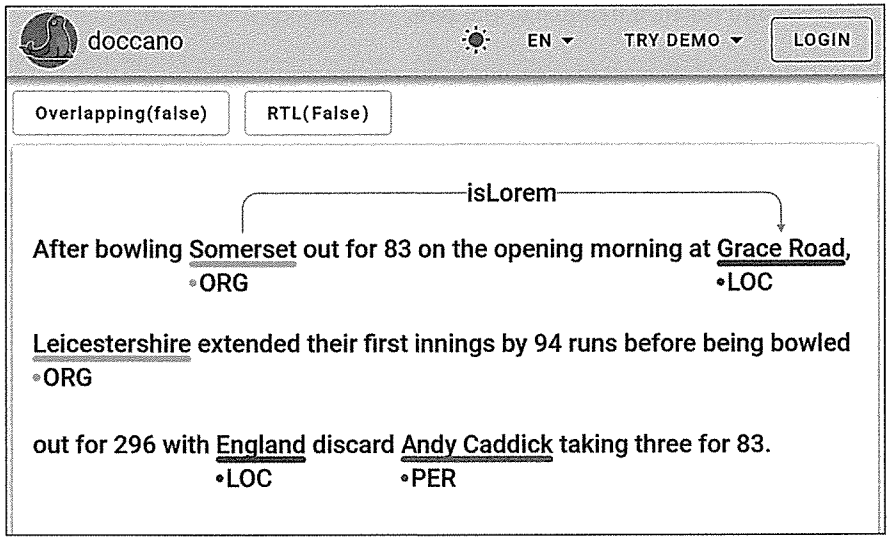


Figure 15. Example of organizations, locations, and people annotated in Doccano. Screenshot is from the official Doccano "demo" site (http://doccano.herokuapp.com/).

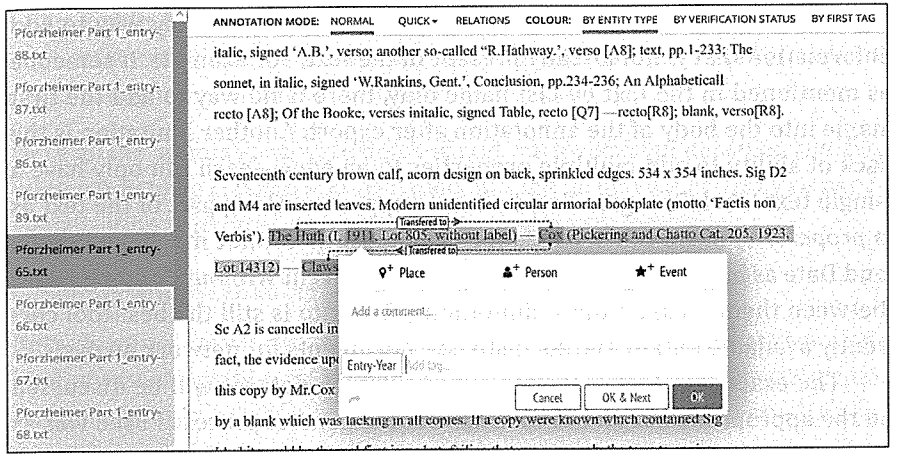


Figure 16. Example of textual annotation in Recogito. Recogito allows a user to customize tags and to relate annotations to one another.

for example, one could ask a computer to find everything that looks like a date in the format MM/DD/YYYY. Machine learning methods from Named Entity Recognition can automatically recognize names of people, places, and organizations. However, machine learning methods require custom training to be reliable, and training a model necessitates manual annotation anyway, so for smaller projects it makes more sense to do all annotation by hand. Manual annotation also comes with benefits—one gets to know the data very well, and it is easier to locate edge cases, quirky bits of data that do not quite fit expectations. It is in the annotation process that one can see how close the data model is to fitting the reality of the data and where changes need to be made. Many of the tools for labeling text are designed for creating training data for machine learning algorithms. Popular tools include Doccano, Prodigy, and Brat, all of which allow users to highlight a portion of text and assign a label to that selection (Figure 15).

For network analysis, we need the ability to not only label text but to capture the relationship between pieces of information in our annotations. And because historical data is often fuzzy, it is helpful to be able to assign multiple labels to the same piece of text—for example, to label a name that we are not sure of as both "person" and "uncertain." While some of the popular tools do have relationship-drawing capabilities, most do not allow multiple tags on the same text in this way. The humanities-focused tool Recogito stands out in a field of tools made for designers of machine learning models. Not only can is it easy to draw and label relationships between labelled text, but a single piece

One substantial limitation of Recogito is there is no easy way to add information that is not already present in the text. For example, if someone is mentioned in the text by last name only, there is no way to add the first name into the body of the annotation after export. Another limitation is the lack of ability to add multiple properties to an edge—each can only have a single text label. Recogito also does not support marking part of the text as a *property of* some other part—for example, one cannot mark the Person and Date as both being properties of the same Event without drawing edges between them. Despite these limitations, Recogito is still the best tool currently available for annotating historical documents for network analysis.[17]

The exact annotation strategy used for each dataset will be discussed in the appropriate chapter to follow,[18] but the general process I used was as follows:

1. Upload the text files that resulted from Segmentation to Recogito in batches of twenty. Recogito stores each set of files uploaded simultaneously as a single "document" with multiple parts. Uploading in batches rather than all at once (or, conversely, one at a time) allowed me to work a reasonable number of files in one sitting and to easily remember where I left off between annotation sessions. Because this would take a long time to do manually and Recogito has no public API, I wrote a browser automation script to upload batches.
2. Annotate according to the requirements of the data model for each case study. To circumvent the "text property of other text" limitation in Recogito, I nested labels. For example, in the Cotton loan records, I highlighted the entire text of a loan and applied the label "Loan," then selected and labelled the Person and Date within that loan separately.
3. After all documents were annotated, I used a web automation script to export annotations and relationships for all the batches of documents as CSV files, then combined those files for all batches into a single annotation and single relationship file.

17 Note that the newly-released "Recogito Studio" uses some of the same underlying technology as the original Recogito, "Studio" lacks the relationship-tagging and export capabilities necessary for annotating network data. The version of Recogito used in this project is, at time of publication, hosted by the Pelagios Network at https://recogito.pelagios.org or available for self-hosting from its Git repository http://github.com/pelagios/recogito2.

18 Annotation was only necessary for the Cotton and Pforzheimer data sets, because

4. Finally, I wrote a script to parse overlapping annotations into *groups* of annotations, re-associating the information that the limitations of Recogito forced me to dissociate.

Once all the annotations were finished and retrieved from Recogito, the data were ready to be cleaned and reshaped into the format expected by network visualization software.

Cleaning and Reshaping

Data cleaning is the process of taking "raw" data straight from a tool or some kind of transformation and tidying it up: completing incomplete fields that were missing in markup, regularizing the format, correcting spelling (as appropriate), removing duplicates, and generally correcting errors. Data cleaning and transformation is usually necessary after using OCR to extract text, or when transferring data from one tool to another. Normalization specifically, a subset of data cleaning activities, is both useful and challenging when working with historical data.

In this project and others like it, the two most challenging types of information to work with are names and dates. On the one hand, normalizing the spelling of names is immensely useful when working with those names in a computational setting—it is easiest to search for all appearances of a person, for example, if their name is spelled only one way. On the other hand, because there are often *not* standardized spellings in early historical periods, the task of deciding on one spelling to replace the rest falls to the modern researcher. Dates pose a different challenge, more to do with uncertainty and levels of precision. Historical data often contains dates with varied levels of specificity—one event may be associated with a full month, day, and year; another may have only a year; another may have definitely happened in July, but *which* July is anyone's guess. Cleaning is usually necessary to bring a level of consistency to data that enables the analysis that will follow.[19]

Cleaning always involves omission. If we completely normalize spellings and names, we omit some of the nuance included in the original source; if we choose to meticulously replicate what was seen in the data source, we obscure messiness of the process that got us there. In the context of this

19 Some scholars argue against the notion of "cleaning" data altogether—Rawson and Muñoz in "Against Cleaning" object to the notion that data is "dirty;" Cordell in "'Q I-Jtb the Raven'" is interested in what might arise from taking dirty data seriously. While I agree with criticisms of data cleaning in some contexts, there are many cases

project, because I am primarily interested in the *idea* of the person, not the way they are represented in the documentation (the spelling, whether they are referred to only by a title or surname, etc.), I have chosen to normalize names.

There are many ways to do data cleaning, ranging from the highly manual to the entirely automatic. If the cleaning that needs to be done is mostly about getting the data into a different *shape*—changing the date format from mm/dd/yyyy to yyyy-mm-dd, removing unneeded data fields, or splitting names into First and Last, for example—the process can be automated using scripts in Python, R, or other programming languages. On the other side of the spectrum, if there are many decisions that need to be made about individual data points—exactly which version of a name to keep, for example—a more hands-on approach might involve carefully going row by row through a spreadsheet representation of the data to make changes individually or through the use of spreadsheet formulas. The time this takes scales as your dataset gets bigger, and at a certain point it becomes impossible to do this entirely by hand.[20]

Scripting data cleaning is challenging and time-consuming, while spreadsheets alone are limited in power. The best tool available for the task is a beautiful blend of the two called OpenRefine. OpenRefine is the superhero version of spreadsheet software—ss data is still presented in a spreadsheet-like table format, but the powers of transformation available are much greater. Instead of editing individual cells, OpenRefine encourages editing *similar* cells en masse. Among its most powerful capabilities, and well-suited to the normalization tasks required for this project, is the "Cluster & Edit" feature. Using a number of provided algorithms, OpenRefine can show you all similar-looking values in a column and change them all at once to whatever value you specify. This is especially useful for people's names from periods where spelling was less standardized—for example, the clustering tool in OpenRefine will recognize that "Bernard of Clairvaux" (with no "e") and "Bernard of Clairveaux" (with an "e") are very likely the same person. OpenRefine also has features for editing the values of data columns based on values in other columns, splitting and merging multi-valued cells, and other transformations that speed up data cleaning massively.

For my two print-born datasets, the Pforzheimer catalogue and the Cotton Library loan records, I used a combination of scripting and OpenRefine

20 On the other hand, fully automatic cleaning leaves room for errors to enter unnoticed. Balancing the two is one of the challenging parts of managing a data

to clean the results of the annotation I did in Recogito. Because I chose to upload documents in *batches* for annotation, exporting my annotations from Recogito resulted in many individual CSV files which I merged using a custom Python script.

As previously discussed, Recogito does not natively support linking nested annotations—any overlapping entities are treated as independent. As part of my annotation strategy, I used one tag as a wrapper encompassing the whole loan or provenance string and nested the individual pieces of information *about* that event (the date, the person's name, the book, etc.) as separate labelled entities within the wrapper. Because Recogito treats all those as independent entities, once they are exported they have to be re-associated with one another. Recogito does include information about where in the text each tag starts and ends, which can be used to programmatically make those connections in the form of a "group ID"—a unique number to identify a cluster of entities that go together. After identifying groups of annotations, I uploaded the data into OpenRefine to combine all the rows in a group into a set of columns instead, normalize name spellings, and assign unique identifiers to people to ensure that if there were five people all called John Smith, they would be easily differentiated. The specific transformations applied to each dataset are discussed in their respective chapters.

When data cleaning is finished, the result is two separate CSVs: one with all the information about the nodes in the network, and one with all the information about their relationships. This is in alignment with the expectations Gephi's Spreadsheet input type, discussed during the Planning section. The CSVs of cleaned data are ready to load into Gephi for analysis, ready for others to reuse for their own analyses or to be transformed into any number of derivative datasets that centre different aspects of the data.

Visualization and Analysis

Finally—after planning, modelling, scanning, OCRing, segmenting, annotating, and cleaning—the data can be visualized. Once data is loaded into a Gephi workspace, a number of graph layouts and visual style adjustments can be made, statistics calculated, and sub-visualizations produced. In this section, I discuss the visualization features I make use of most in the chapters to follow.

Edge Weights

Loading data into Gephi is, in some sense, the last step of the data cleaning process. Gephi's importer handles a few tasks to ensure valid network structures, including automatically removing duplicate nodes from the Nodes table and merging parallel edges (those that run between the same two nodes). If parallel edges are detected and the edges do not already have a "weight" value (essentially, how strong the connection between two nodes is), Gephi will automatically calculate and insert the weights for every edge in the network. Edge weight can contribute visually to a network by changing the thickness or colour of the lines between nodes, and by affecting the layout of a network graph when using certain algorithms—a point I will return to shortly. Edge weight also plays into network statistics that one can calculate and use for analysis or as variables in visualization.

Network Statistics

The statistical side of network analysis, describing the characteristics of a whole network or nodes within those networks numerically, can be used as a component of visualization. There are many metrics mathematicians use to describe graphs with varying degrees of complexity—PageRank, for example, is a complex metric used by some search engines that models the internet as a network of linked pages to determine how high a certain page should appear in search results. For the purposes of my case studies and similar projects, the two most important metrics to understand are degree and betweenness.

Degree and betweenness are both measures of a single node's *centrality* in a network. Centrality is what it sounds like—how central, important, integral a node is to its network. Degree measures how many edges a node has attached to it. In directional networks, out-degree measures the number of edges leaving *from* a node, while in-degree measures the number of edges

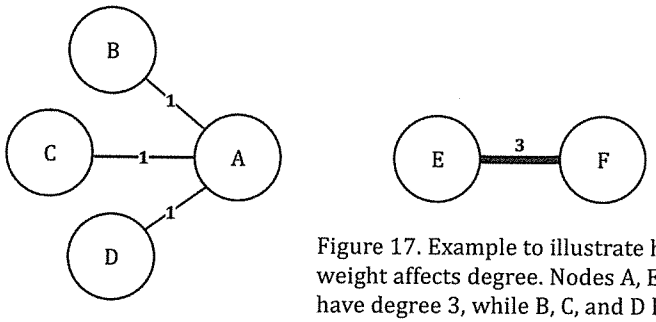


Figure 17. Example to illustrate how edge weight affects degree. Nodes A, E, and F all have degree 3, while B, C, and D have degree 1.

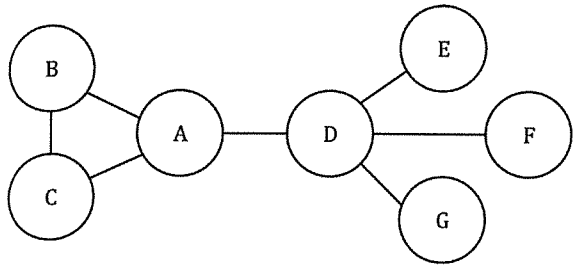


Figure 18. Sample network to demonstrate betweenness centrality. Nodes A and D have high betweenness because they act has "bridges" between the ABC and DEFG clusters. The shortest paths between B/C and E/F/G must pass through A and D.

pointing *to* a node. Weighted degree takes into consideration the weights of the edges—a node with three edges connected to it, each with weight one, would have the same *weighted degree* as a node with a single three-weight edge connected to it (see Figure 17). Degree is one way of measuring how connected a node is to other nodes in the network, based solely on the *number* of connections it has.

Betweenness centrality is another way of measuring how connected a node is in a network—one that tries to emphasize the quality of connections over strict quantity. Instead of counting the number of edges that go in or out of the node, betweenness centrality considers how many of the "shortest paths" between two nodes in the graph run through a given node. Imagine playing a game of "six degrees of separation," trying to link two actors based on their co-stars in various movies—Kevin Bacon was in *In the Cut* with Mark Ruffalo, Mark Ruffalo was in *Iron Man 3* with Jenna Ortega, etc. Someone with a high betweenness centrality might be akin to a random character actor who seemingly appears in *everything* and is thus a common connecting link between other pairs of actors. Someone else with a high betweenness number might *not* be in all that many movies but have appeared in one large-cast comedy and one large-cast drama whose casts did not otherwise overlap much, positioning them as the link between two worlds. In short, betweenness measures how good a node is at being a bridge between other nodes. Nodes with high betweenness centrality help information traverse a network quickly (Figure 18).

Ego Networks

Once a large network is loaded into Gephi, subsets of that network can be created without having to create new lists of nodes and edges using Filters. Filters can be applied to a network to hide certain nodes or edges according to their metadata attributes or statistics. As with many of Gephi's features,

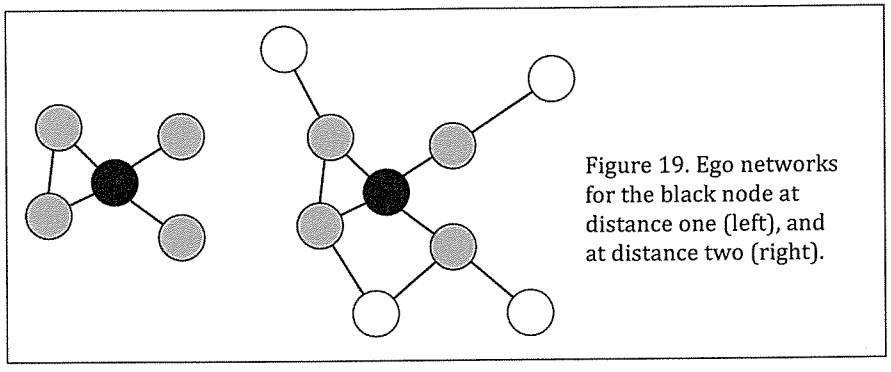


Figure 19. Ego networks for the black node at distance one (left), and at distance two (right).

ful is the "ego network" filter. Ego networks are networks that contain only the nodes that are connected to a single, central node. Ego networks have a property called "distance" that indicates how many degrees removed a node can be from the central node and still be included in the network. For example, at a distance of one, your ego network includes you and all your friends; at a distance of two, your ego network includes you, your friends, and your friends' friends. Ego networks can help narrow in on a specific node and understand its place in a very large network by removing some of the visual noise caused by a high number of nodes (Figure 19).

Layout Algorithms

The component of network visualization that has perhaps the most drastic effect on the appearance and interpretation of a network graph is the layout—the arrangement of nodes relative to one another in the space.

The layout algorithm I use most commonly in the case studies to follow is ForceAtlas2 (FA2). FA2 is a force-directed layout; nodes want to repel each other like mismatched magnets, but are pulled together based on the weight of the edges between them. All nodes start in the centre and are given a nudge in a random direction, then the edges between them change their trajectories and pull nodes together. How strong the pulls between nodes are is adjusted through a parameter called "gravity"—nodes behave a bit like a system of planets interacting with one another.[21] Because nodes' initial trajectories are determined by a random nudge, FA2 will not produce exactly the same layout every time it is run.

21 For more details on the FA2 algorithm, see Jacomy et al., "ForceAtlas2, a Continuous Graph Layout Algorithm for Handy Network Visualization Designed for

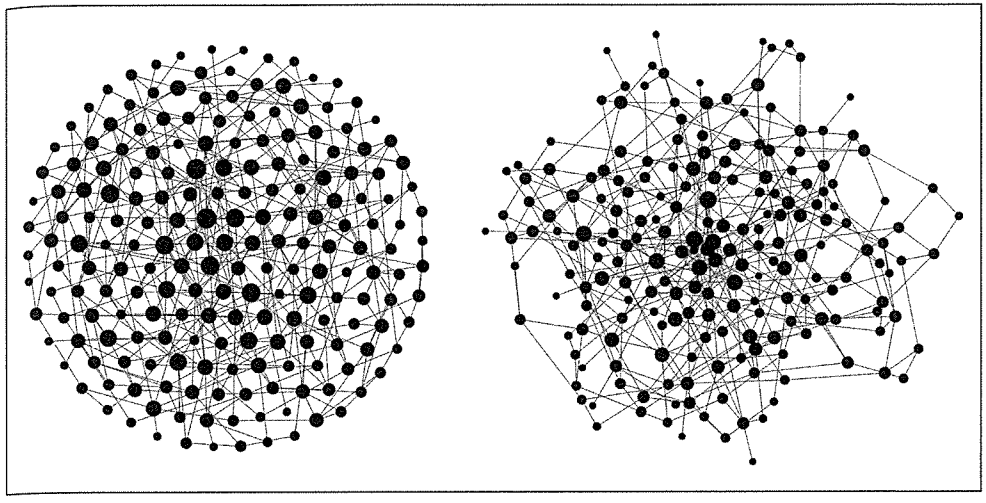


Figure 20. The same network arranged according to the Fruchterman Reingold layout (left) and Force-Atlas2 (right).

The primary appeal of ForceAtlas2 as a layout algorithm is how apparent it makes clusters, groups of nodes that are strongly connected to each other but barely (or not at all) connected with nodes outside their group, like a school clique (Figure 20). FA2 tends to push nodes with few connections out toward the edges of the network and pull the most highly connected nodes toward the centre. In networks that are densely connected, this can mean that the centre is hard to make sense of visually, but that difficulty can be overcome using the other visualization components above to filter or otherwise make elements of the network visually distinct. FA2 and other force-directed algorithms convey meaning through the spatial arrangement of nodes.

Because FA2 uses space to convey meaning, it can be easy to assume that proximity in space *always* corresponds to proximity in the *structure* of the network. However, this is not the case when nodes have competing forces acting on them. For example, a node connected with equal weight to two otherwise disconnected clusters will wind up halfway between the two and may be coincidentally physically close to nodes that it has no connections to. In Figure 21, which provides a visual representation of this phenomenon,[21] node D is pulled between nodes A and H. The tight clustering of nodes EFG and HIJK and the proximity of node B to node D leads to the appearance that D could be part of a central ABCD cluster, when, in reality, D is as connected to B as it is to I and E. This effect is amplified in larger, more complex

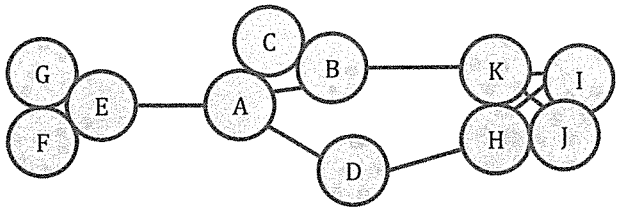


Figure 21.
Example of a force-directed layout resulting in potentially deceptive visualization. D appears physically close to the ABC cluster, but is mathematically equally close to nodes E, B, and I.

network visualization—the ability to actively select, zoom, filter, and otherwise manipulate a network graph is a key guard against misinterpretation of layouts.

Colour and Size

If the nodes and edges in a network have metadata beyond the bare minimum "identifier and label" demanded by Gephi—such as additional tags added during data processing or statistics calculated by Gephi—that metadata can be used to change the appearance of the network. Nodes and edges can be sized or colour-coded based on discrete values—such as a label for "type" (book, person, text, etc.)—or continuous values—such as degree or date. Colour and size add dimensions of visual communication beyond the node-edge structure to a network—they can be used to convey additional information not indicated by the network structure itself.

One very important consideration in designing data visualizations is colour accessibility. Especially when there are many variables in play, it can be tempting to use nuanced colour coding in network diagrams. However, randomly selected or generated colour palettes are unlikely to have sufficient contrast for colourblind or low-vision viewers. Most of the default palettes offered in Gephi do not pass muster under simulations of the three most common types of colour blindness. For the visualizations in this project, I have used the colourblind-friendly palettes described by Paul Tol.[22] These or other colour palettes, while not available directly in Gephi, can be added manually using the hex codes for each colour.

Another important consideration is the rhetorical power of size. Making certain nodes or edges in a network larger than others suggests they

are more important, even if the sizing were determined randomly—larger nodes literally take up more space in a visualization and will consequently take up more space in the viewer's thoughts. As with any data visualization, it is possible to intentionally or unintentionally distort the viewer's understanding of the data by using a misleading scale, where the difference between sizes is very large or very small. It is the responsibility of both the creator of visualizations and the viewer of them to use caution when choosing or interpreting scale.[23]

Interactive and Static Exports

Interactivity is an important part of network visualization. Most of the value of networks as a vitalization strategy is in the way they allow a person to navigate an otherwise overwhelming volume of information in a (hopefully) friendly graphical format. Static visualizations like those that appear throughout this document make many assumptions about what will interest the viewer. For large networks, it is not possible to represent the entire network in a single page-width image and have all, or even most node labels readable; in a static image, the user cannot select a node to highlight its connections, zoom in and out, or otherwise explore the network on their own. Static images may be of use for making specific arguments about a network but are not suitable for an approach that views network visualization as a kind of reference work.

By default, Gephi only supports exporting static images. Gephi project files are shareable, if a bit cumbersome, but requiring people to install and learn Gephi in order to interact with network visualizations is not a realistic solution. Thankfully, there are tools for displaying interactive network visualizations in web browsers using JavaScript. As part of a 2012 project called "Interactive Visualizations for Teaching, Research, and Dissemination," researchers at the Oxford Internet Institute (OII) developed a plugin for Gephi to enable exporting the required files to display interactive networks in web browsers using SigmaJS.[24]

SigmaJS allows viewers to select individual nodes to view their associated metadata (like type, date, author, or any other attributes included in the Node table) and ego networks (at distance one). Users can zoom in and

23 For further discussion of responsible data visualization, see Boyd Davis, Vane, and Kräutli, "Can I Believe What I See?"; Drucker, "Graphical Approaches to the Digital Humanities."

out and move around the network as they please. It also includes a search feature for locating nodes in the network, which is an especially valuable feature in large networks. Visualizations exported by the OII plugin do not require any special website configuration to view; they need only to be opened in a browser, making them highly portable and easier to preserve in something like the Internet Archive or a GitHub repository. For this project, I modified the template created by the OII plugin to include a dark mode, display metadata about edges when a node is selected, and additional information in the legend. At time of writing, the visualizations included in the case study chapters that follow are available in interactive form online[25] or for download on GitHub.[26]

Iteration

Visualization often reveals the need for further visualizations, whether because something unexpected in the network stood out and demanded to be looked at more closely, or because something expected was not apparent in the network and required a change in approach. When producing visualizations, I often returned to the Transformation step to produce alternate versions of the Nodes and Edges table to re-visualize the network with additional or differently-grouped data. While much of the data processing system I have laid out in this chapter happens linearly—moving from step to step in sequence—visualization is a highly iterative activity, and one that never truly feels "finished."

25 https://manuscriptnetworks.online/.

PART TWO

Part One has showcased the labour required to work with humanities data and create interactive visualizations that can serve as exploratory tools for scholars. Many of the process and challenges I have outlined here apply not only to networking projects but to all projects using digital methods to work with non-digitized, humanistic materials. In the chapters that follow, I discuss the data challenges specific to each case study, offer a selection of visualizations from each, and propose further work to be done with these and similar data.

Part Two focuses on applying the method described in Part One to three case studies. The origin story of these case studies is as follows:

I first became interested in antiquarianism when investigating the provenance of a particular little book in the Stanford University Libraries' Special Collections—a copy of Matthew Parker's 1566 *A Testimonie of Antiquitie*, an edition of an Old English homily on the sacrament and the first printed Old English text.[1] The Stanford University Library copy is bound in a limp vellum fragment of a twelfth-century manuscript of Acts of the Apostles, on which several previous owners have written—one "Richard Ballett" and one "Broughton Lib." In trying to determine who the one-time owner "Broughton" might be, I came across the writings of a Richard Broughton in a collection of discourses from a meeting of the Elizabethan Society of Antiquaries, a group of which Sir Robert Cotton was a founding member. This led to an interest in Cotton and how he lent books from his library to his fellow antiquaries. I hoped to find Richard Broughton among those who borrowed books from Cotton, and that any books he borrowed might give me a sense of his interests and suggest whether he would be the kind of person to own a copy of the *Testimonie*. While it turned out Broughton is not among the Cotton borrowers, I wondered what might come out of studying patterns in relationships between the Cotton books and those who borrowed them. This question clearly lent itself to a network model—nodes of people, nodes of books, and relationships between them.

When the COVID-19 pandemic hit and access to primary sources was no longer possible, I decided to pursue this project. As I dove into learning about network analysis and discussed what I learned with friends and colleagues, I began to see possible applications of network analysis everywhere. One colleague's study of fragmentation and miscellany led me to the Digital Index