UNIVERSITY OF OKLAHOMA

GRADUATE COLLEGE

VARIANTFLOW: INTERACTIVE STORYLINE VISUALIZATION USING FORCE DIRECTED LAYOUT

A THESIS

SUBMITTED TO THE GRADUATE FACULTY

in partial fulfillment of the requirements for the

Degree of

MASTER OF SCIENCE

By

SHEJUTI SILVIA
Norman, Oklahoma
2016

VARIANTFLOW: INTERACTIVE STORYLINE VISUALIZATION USING FORCE DIRECTED LAYOUT

A THESIS APPROVED FOR THE
DEPARTMENT OF ENGINEERING

BY

Dr. Christopher E. Weaver, Chair

Dr. Mohammed Atiquzzaman

Dr. Charles D. Nicholson

Contents

List of Figures

Abstract

The study of literature is changing dramatically by incorporating new opportunities that digital technology presents. Data visualization overturns the dynamic for literary analysis by revealing and displaying connections and patterns between elements in text. Literary scholars compare and analyze textual variations in different versions of a lost original text and work to reconstruct the original text in the form of a *critical edition.* A critical edition notes textual variations in extensive footnotes, collectively called a *critical apparatus.* Information in the apparatus is of great interest to scholars who seek to explore complex relationships between text versions. Motivated by application to classical Latin texts, we adapted the storyline technique to visualize a critical apparatus. The visualization facilitates guided discovery of similarities and dissimilarities between prior text versions, which are difficult to detect and reason about with traditional deep reading and spreadsheet-based methods.

Storyline visualizations help users understand and analyze the interactions between entities in a story and explore how entity relationships evolve over time. Typical design considerations in existing storyline techniques include minimizing line crossing and line wiggling, which are computationally intense problems. Generating storyline layouts in real time is a substantial challenge to interactive visualization. Existing storyline techniques support limited user interaction due to the high cost of layout. We contribute a force directed layout algorithm that dynamically reflows storyline layouts with best effort response to internal and coordinated interactions. We anticipate that the characteristics of our layout algorithm will allow for graceful response to a wide variety of interaction types, speeds, and patterns. We conducted a user study to evaluate the legibility of our storyline layout after convergence. The evaluation results demonstrate that most users can accurately complete a wide variety of visual metaphor interpretation, reading, and pattern recognition tasks within 20 seconds.

Chapter 1

Introduction

Textual criticism is an active area of research in digital humanities. Literary scholars study the creation, alteration and evolution of text over time. Data visualization overturns the dynamic for literary analysis by revealing and depicting connections and patterns between elements in text. Interactive visualization techniques guide scholars to explore text corpora and discover new knowledge that is impractical to gain using traditional methods.

Original Latin texts from the classical era are mostly lost. Only replications of the original texts survived in the forms of manuscripts, early printed editions, and modern editions. Existing manuscripts and the printed editions of a text usually vary from each other significantly, due to the errors or alteration of text introduced by scribal and printing processes. Classics scholars refer to these manuscripts and early printed editions as *witnesses*. Textual differences between witnesses are called *variants*. A scholar's ultimate goal is to reconstruct the original text, in the form of a *critical edition*, by carefully choosing variants from the existing witnesses. The main text in a critical edition is accompanied by a *critical apparatus*, which includes variants from witnesses that the editor considered important during *collation*. Each word or phrase in the main text that has an entry in the apparatus is called a *lemma*. The apparatus appears at the bottom of each page, in a highly abbreviated form that requires substantial training to read. Students and novice readers of Latin texts are typically unable to read or interpret the apparatus.

This thesis presents VariantFlow, an interactive storyline visualization of the critical apparatus that is easily readable and comprehendible to students, teachers, and novice readers of Latin texts, who do not have sufficient training to read the apparatus in its canonical form. Moreover, scholars are unable to analyze the apparatus of an entire critical edition at once. They are only able to look at an individual entry or a collection of entries in an apparatus on a page, instead of the entire text. VariantFlow visualization provides an overview of the entire apparatus that will help scholars observe larger patterns and anomalies, above the level of the individual lemma. The storyline layout works especially well for tracing witnesses of the same lineage (their *stemma*) that have many variants in common, traveling through text together, revealing similarities and dissimilarities between them. Analyzing the apparatus with VariantFlow visualization will also help scholars understand the editor's conjectures in an existing collation, which plays an essential role in creating new and improved critical editions.

State-of-the-art storyline techniques compute optimal layout with minimum line crossing and line wiggling. These techniques support limited user interaction due to high computational cost of layout. Instead of generating an optimal layout, we focus on generating a layout that is highly interactively responsive yet also topologically good enough for users to perform key reading and pattern recognition tasks with reasonable speed and accuracy. We use a customized force-directed layout algorithm to achieve fast convergence to reasonable layouts regardless of earlier layout states or interaction dynamics. We are developing the new storyline technique to be a component in an upcoming desktop app for visualizing critical editions in the Digital Latin Library. One mode of the app will include brushed highlighting and filtering between storyline elements and the elements in a central view of the main text.

Existing storyline techniques employ combinatory optimization methods that are difficult to apply incrementally and hence do not accommodate interaction well. In contrast, our layout algorithm runs smoothly and continuously in response to interaction, and hence also seems faster (while actually being slower) than existing techniques. We argue that these characteristics allow for graceful response to diverse forms of coordinated interaction, such as dynamic filtering. A user evaluation confirmed that our storyline layout is aesthetically pleasing and easy to read for novice readers, who performed several reading and pattern recognition tasks with accuracy and efficiency not significantly affected by layout complexity.

This thesis contributes ***a storyline layout algorithm for VariantFlow visualization, that dynamically generates legible layout for best effort response to external interaction.*** An interactive storyline generation technique needs to be able to relayout within 100 ms of an external interaction [26]. In the best case scenario (with a smaller data set), our layout is generated in interactive time (within 100 ms), however for the worst case (with a larger dataset), it can take up to several seconds. We have employed a force-directed layout technique to achieve fast convergence to reasonable layouts regardless of earlier layout state. ***We hypothesize that the characteristics of our force-directed layout algorithm will allow for graceful response to a wide variety of interaction types, speeds, and patterns.***

This document opens with a discussion of the application domain area and the related work in both humanities and data visualization that motivated our research. In Chapter 3 we describe our objective, approach and challenges in designing VariantFlow, an interactive storyline visualization. Chapter 4 details the VariantFlow layout algorithm, and its time complexity, implementation, and limitations. Chapter 5 discusses the procedure and results of a usability evaluation of the prototype visualization. In Chapter 6 and 7 we conclude by considering utility and usability of our prototype layout algorithm and visualization, plus several future directions.

Chapter 2

Background

This thesis, like much work in visual analytics, draws techniques and terminology from a variety of disciplines: *text visualizations*, *time series visualizations*, *storyline visualizations* and *force-directed layout algorithms* come from data visualization; *textual variants*, *critical apparatus*, *critical edition* and *textual collation* comes from classical Latin editorial work; *variant graphs* and *collation tools* come from digital humanities. This chapter presents related work and provides background in these areas to aid understanding the motivation behind the design of the VarientFlow visualization.

2.1 Data Visualization

Data visualization is the theory and practice of creating visual representations of data that enable users to gain new insights and discover hidden patterns in a given dataset. Application of data visualization includes various domain areas, such as intelligence, social networks, sports, science, and the humanities. *Text* is the most common form of "data" used in the humanities. Standard text visualization techniques, focusing on the analytics of large datasets, are usually not tailored to serve the needs of the humanities scholars who perform critical engagements with texts. Few existing visualization techniques depict complex relationships between text entities. On the other hand, *time series* and *storyline visualization* techniques are commonly used to represent entity relationships in any given dataset and how entity relationships change over time. Due to salient issues in the humanities—uncertainty, interpretative complexity, and the idiosyncrasies in critical theoretical approaches—it is difficult to develop visualization tools to support humanities research. In the next section, we present the specific humanities domain area and the complex research problems within that domain area that our research aims to support.

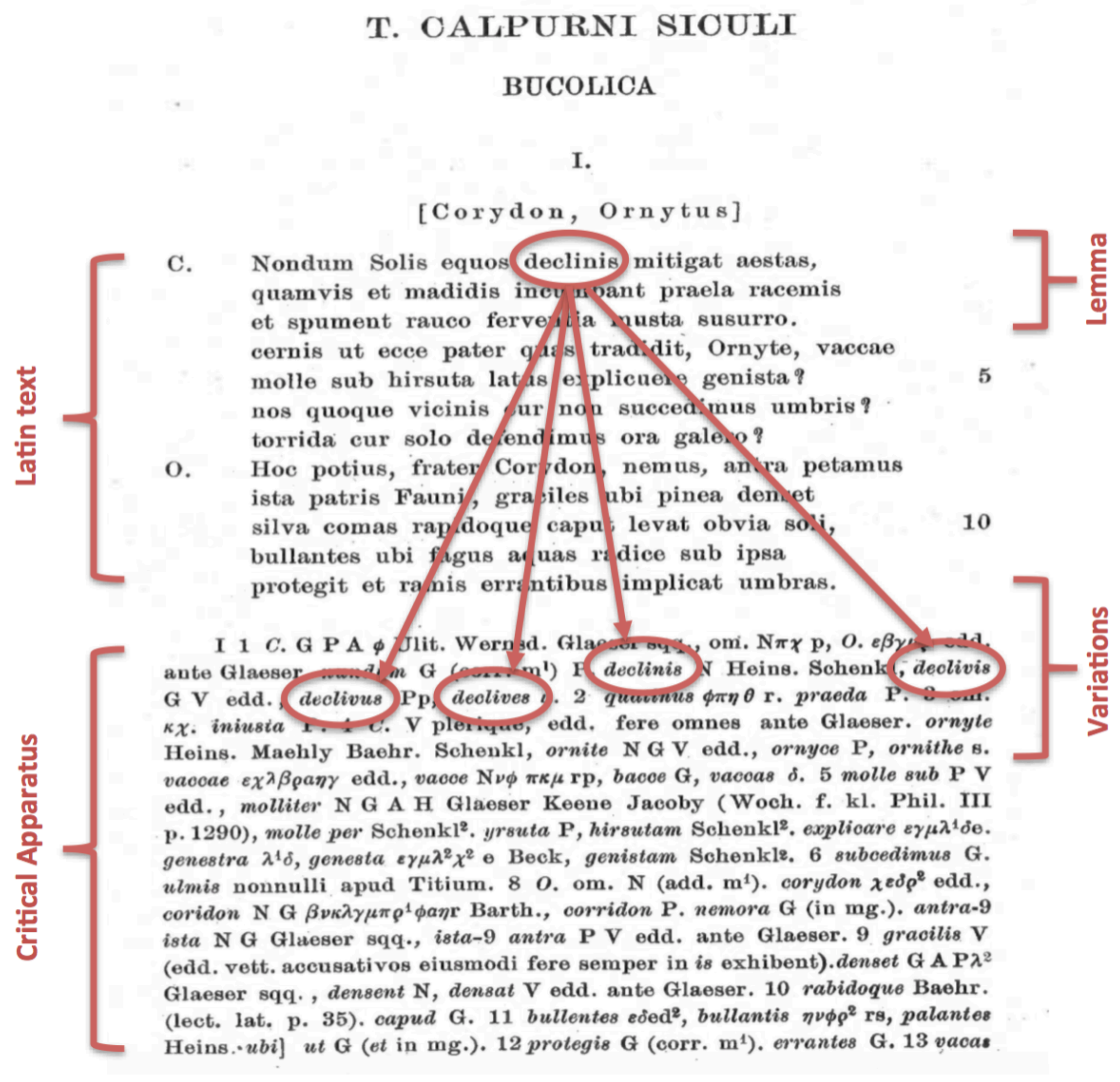
2.2 Application Domain: Classical Latin

Prior to the 15th century, Latin texts were primarily preserved by scribes, who copied the original text or a copy of the original text to a manuscript by hand, introducing textual variations and transcription errors in the process. Over time, the "original text" is replicated to multiple manuscripts that vary from each other in ways that are important to literary scholars and textual critics. At the end of the 15th century, publishers started publishing printed editions of these manuscripts. Publishers often chose a manuscript at random to publish as a printed edition, and then discarded the manuscript itself, considering it redundant to the printed edition. Thus, manuscripts of many classical Latin texts are lost. Moreover, the printing process itself introduced errors [16, 21, 33]. As a result, existing manuscripts and the printed editions of a lost "original text" usually vary from each other significantly. Manuscripts and early printed editions of an “original text” are called *witnesses*. Textual variation between witnesses are called *variants*.

Scholars and editors of Latin texts study the transmission history and evolution of texts to reconstruct the original text as closely as possible. A scholar’s ultimate goal is to produce a *critical edition* that recreates an original text from carefully selected variants. First, the editor chooses the most reliable *witness* as the *original text*, also referred to as the *base text*. Then he selects a set of witnesses that he would like to *collate* against the base text. During the collation process, the editor manually compares witnesses with the base text and records variants for each word, typically in a spreadsheet, referred as a *collation table*. Then the editor employs philological judgment and his own reasoning to determine the authenticity and the significance of each variant in the collation table [4, 16, 21, 33].

Editors take various approaches to select variants for critical editions. They often prefer variants supported by the majority of witnesses, or the oldest witnesses, or the witnesses that consistently provide reliable variants throughout the text. In some cases, editors consider the *outliers*—the variants that deviate from the known practice of a scribe or a given period—more reliable [3, 16, 21, 33].

Editors also employ a *stemmatic approach* to select variants for critical editions. A *stemma* is a family tree of manuscripts showing which manuscripts copied from each other. This approach requires tracing the relationship between witnesses from one generation to the others. Depending on the witnesses sharing the same lineage, and therefore sharing similar variants, the editor creates a manuscript family tree (stemma). This helps to reduce the complexity of the collation process by allowing the editor to select variants for the critical edition from a branch that he or she thinks more reliable, and ignore variants from the other


Latin text
Lemma
Variations
Critical Apparatus

T. CALPURNI SICULI

BUCOLICA

I.

[Corydon, Ornytus]

C. Nondum Solis equos declinis mitigat aestas,
quamvis et madidis incumbant praela racemis
et spument rauco fervantia musta susurro.
cernis ut ecce pater quas tradidit, Ornyte, vaccae
molle sub hirsuta latus explicuere genista?
nos quoque vicinis cur non succedimus umbris?
torrida cur solo defendimus ora galero?
O. Hoc potius, frater Corydon, nemus, antra petamus
ista patris Fauni, graciles ubi pinea denset
silva comas rapidoque caput levat obvia soli,
bullantes ubi fagus aquas radice sub ipsa
protegit et ramis errantibus implicat umbras.

I 1 *C.* G P A *φ* Ulit. Wernsd. Glaeser sqq., om. N*πχ* p, *O.* *εβγ* edd. ante Glaeser. *nondum* G (corr. m¹) P. *declinis* N Heins. Schenkl, *declivis* G V edd., *declivus* Pp, *declives* ω. 2 *quamvis* *φπηθ* r. *praeda* P. 3 om. *κχ*. *iniusta* P. 4 *C.* V plerique, edd. fere omnes ante Glaeser. *ornyte* Heins. Maehly Baehr. Schenkl, *ornite* N G V edd., *ornyce* P, *ornithe* s. *vaccae* *εχλβρaηγ* edd., *vacce* N*νφ* *πκμ* rp, *bacce* G, *vaccas* *δ*. 5 *molle sub* P V edd., *molliter* N G A H Glaeser Keene Jacoby (Woch. f. kl. Phil. III p. 1290), *molle per* Schenkl². *yrsuta* P, *hirsutam* Schenkl². *explicare* *εγμλ*¹*δ*e. *genestra* *λ*¹*δ*, *genesta* *εγμλ*²*χ*² e Beck, *genistam* Schenkl². 6 *subcedimus* G. *ulmis* nonnulli apud Titium. 8 *O.* om. N (add. m¹). *corydon* *χεδρ*² edd., *coridon* N G *βνκλγμπρ*¹*φaη*r Barth., *corridon* P. *nemora* G (in mg.). *antra*-9 *ista* N G Glaeser sqq., *ista*-9 *antra* P V edd. ante Glaeser. 9 *gracilis* V (edd. vett. accusativos eiusmodi fere semper in *is* exhibent). *denset* G A Pλ^2 Glaeser sqq., *densent* N, *densat* V edd. ante Glaeser. 10 *rabidoque* Baehr. (lect. lat. p. 35). *capud* G. 11 *bullentes* *εδed*², *bullantis* *ηνφρ*² rs, *palantes* Heins. *ubi*] *ut* G (*et* in mg.). 12 *protegis* G (corr. m¹). *errantes* G. 13 *vacas*

Figure 2.1: Giarratano's critical edition of the classical Latin poem *Calpurnius Siculus* [29].

branches in the stemma that the editor considers less reliable [3, 16, 5, 21, 33].

Finally, the editor assembles the *collated text* as a critical edition. Figure 2.1 shows Giarratano's critical edition of the classical Latin poem. As shown in this figure, the text in a critical edition appears at the top of the page, accompanied by a *critical apparatus*, which includes the list of witnesses and variants the editor considered important during the collation process. An apparatus consists of entries for each *lemma*, meaning word or phrase in the text, for which there are textual variations among witnesses. The apparatus appears as a running set of footnotes on each page, containing a collection of entries for the lemmas associated with that page. Each entry in the apparatus contains the line number and the variants for a lemma, followed by the type of each variant and the list of witnesses that supports that variant. The information in the apparatus is encoded in extremely concise and highly abbreviated form that requires substantial training to read. Students and novice readers of Latin texts are typically unable to read or interpret an apparatus.

We developed the VariantFlow visualization to represent the information in a critical apparatus in a way that requires much less training to read and comprehend. In the apparatus, editors only include the conjectures and the collation decisions that they consider important. Thus, visualization of an apparatus reveals the key editorial decisions made in a critical edition, and the patterns of witnesses and variants that are reflected in that edition. For scholarly advancement, being able to read and understand the apparatus using a visualization tool is quite valuable to the students, teachers, and novice readers of Latin texts, who do not have sufficient training to read the apparatus in its canonical form.

VariantFlow visualization can support editors to assess their own editorial or collation process over time. Editors take years to create a critical edition. Over the years, editors build their collation table, starting from the beginning of a base text and typically working their way through to the end of the text. How editors construct the collation table, their interest in specific characteristics or patterns in the texts they compare, and which editorial decisions and philological judgments they make for any given conjecture or any given lemma in the critical edition, can all vary over time. Therefore, visualization of a critical apparatus, which includes the key phenomena of the collation process, can usefully reveal substantial “drift” in editorial and scholarly decisions, from chapter to chapter or from the beginning of the text to the end. For example, when the editor collates variants for the first chapter of a text, he may prefer variants from a certain group of witnesses over the others. However, a few years later, when the editor works on the later chapters of the text, he might prioritize a different group of witnesses, due to the change in knowledge, experience, and reasoning. VariantFlow effectively depicts these “scholarly drifts” throughout the text of a critical edition, allowing the editors to study their own collation process in the past.

Similarly, scholars cannot analyze the entire apparatus of a critical edition at once. They can only look at an individual entry or collection of entries in an apparatus on a page. VariantFlow visualization provides an overview of the entire apparatus that helps scholars observe larger patterns and anomalies in variant texts, whereas before, scholars only could see details at the level of a few individual lemmas. VariantFlow layout works especially well for tracing witnesses from same lineage (stemma) that have many variants in common, revealing similarities and dissimilarities between them. Analyzing the apparatus with VariantFlow visualization will also help scholars understand the editor’s conjectures in an existing collation, which plays an essential role in creating new and improved critical editions. Moreover, VariantFlow reveals the structure of the stemma used in creating a critical edition, by grouping witnesses from the same lineage together. It can also depict interesting anomalies like contamination, in which a witness is linked to multiple lineages.

2.3 Related Work

This section presents the related topics in both humanities and data visualization that inspired our research.

2.3.1 Collation Tools

For centuries, textual critics and editors have been comparing texts and their provenance of copying to reconstruct original texts that are lost. Many digital humanities projects are currently invested in developing tools to facilitate this process. *Juxta* is an online open-source tool widely used by scholars to visualize textual differences in multiple witnesses and collate them to create new critical editions [1]. Schmidt introduced *Variant Graph* [25], which represents textual variations as separate paths through a directed acyclic graph, with witness labels and variants positioned on the edges. Due to convoluted layout and the difficulty of effective text positioning, variant graphs have generally poor readability and scalability. Inspired by Schmidt's work, Dekker [9] proposed *CollateX*, a modified version of *Variant Graph*, most commonly used by scholars for analyzing variants. *StemmaWeb* [4] is another online application that extends *CollateX* and provides various methods of analyzing and interpreting textual variants. *TRAViz*, an interactive implementation of variant graph, aligns sentences from witnesses, based on their similarity in tokens (words or phrases) [13]. Thus, *TRAViz* is effective for application cases like verse-by-verse comparison between Bible editions.

2.3.2 Text and Time Series Visualizations

This thesis is closely related to research on text and time series visualization. We are interested in identifying complex patterns in witnesses related to phenomena such as scholarly drift, contamination, and characterizing how these patterns evolve throughout the text of a critical edition. Many existing techniques combine text visualization and time series visualization techniques to depict temporal patterns in text data. ThemeRiver is a visualization of theme changes over time in a document collection. Each theme is represented with ‘river currents’ made of smooth continuous curves. River currents progress through time, from left to right, revealing various temporal patterns in a text collection [12]. History flow visualization shows how collaborative documents change over time, revealing interesting patterns in authorship and textual variations in different versions of a document [30]. TimeNets is a visualization technique for genealogical data that represents individuals with lines that converge and diverge based on various temporal and family relationships such as - birth and death, marriage and divorce, etc. [17]. Other similar visualization techniques track changes of topics in text streams [7, 18, 10, 19], and of creation and evolution of communities in social networks over time [24]. Popular stack graph and layer graph visualization techniques are also effective in revealing temporal patterns and entity relationships in a given dataset. Each stack or layer represents an entity such as topic, genre and revenue of movies. The directed flow from left to right indicates how various entity relationships change over time [6, 10, 19].

2.3.3 Storyline Visualizations

Munroe popularized the storyline visualization technique in hand-drawn form in his XKCD comic about movie narratives [22]. Figure 2.2 shows some hand-drawn illustrations of XKCD’s “Movie Narrative Chart”. In this illustration, the horizontal axis represents time and vertical groupings of lines indicate which

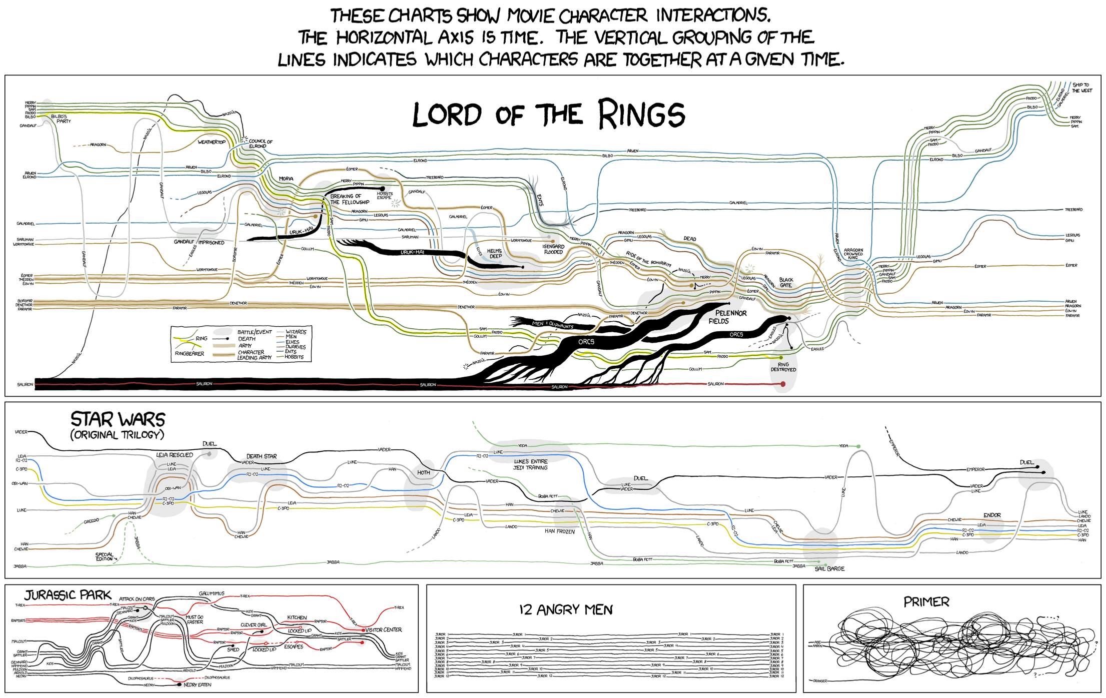


Figure 2.2: XKCD's hand drawn illustration of "Movie Narrative Chart" [22]. (Reproduced under Creative Commons Attribution - Non Commercial 2.5.)

characters are together at a given time. Colored regions reference the spatial locations the characters pass through.

Thus, storyline visualizations represent entities with lines. Lines flow from left to right, converging or diverging from each other at various points in time, revealing spatial, temporal, and correlated entity grouping relationships. For our application case, we are particularly interested in tracing patterns of variations between witnesses and tracking how these patterns change throughout the entire text of a critical edition. The ability to show groupings of lines based on entity relationships and how these groupings change over time depending on changes in entity relationships makes the storyline technique a suitable fit for our application case. For storyline to be effective, the groupings of lines needs to be coherent over time. In our application case, lines represent witnesses, progressing through text (lemmas), from left to right in reading order. Lines are grouped together when they share the same variant for a lemma. It is an assumption within our application case that there is enough consistency between groupings of witnesses in providing common variants throughout the text. Apparatuses usually include witnesses from a few reliable lineages (*stemma*). Witnesses from the same stemma tend to possess many variants in common. Therefore, storyline visualization of the apparatus should effectively reveal patterns. For instance, lines might nicely converge together into a few groups at the beginning of a critical edition, indicating higher quality provenance for those witnesses. Or, towards the end of the critical edition, if there are no coherent groupings in storyline and lines are tangled between groups, it reveals many variations and high *fragmentation between witnesses* (destruction of witnesses, or the presence of fragments of a former witness within other witnesses). Storyline can also depict anomalies like *contamination*, where a line consistently grouped with one group of witnesses throughout the text suddenly diverges and joins another group, revealing its association with multiple lineages.

There is growing interest in automating the layout of storylines [28, 23]. The methods described by Tanahashi, et al. [28, 27] produce storylines for hundreds of entities and event times but take several minutes to lay out, making them too slow for many user interactions including dynamic queries. StoryFlow [20] generates layouts faster, enabling it to support precise user interactions, such as bundling and straightening lines.

2.3.4 Force-Directed Graph Layout Algorithms

Like storylines, graph visualizations use lines to represent relationships between entities. The design principals for aesthetics, legibility and traceability of individual lines in graph visualizations carry over to storylines. Our research was particularly inspired by the design principals and layout algorithms suggested by Ogawa and Ma [23] for generating storylines. In their work, Ogawa and Ma suggested force-directed graph layout algorithms as a potential future research direction for storyline generation. They presented a set of design rules for storyline visualization of temporal dynamics between developers in a software development history. The horizontal axis represents timesteps, and each node represents a developer's appearance in a timestep. Nodes representing the same developer are connected by edges in the horizontal-direction. Thus, each line in a graph represents an entity. Each node on a line is vertically aligned with the co-appearances of the other developers (nodes) associated with the same timestep. By restricting the node positions on their respective timesteps along the horizontal, the force-directed graph layout algorithm can be applied only along the vertical to cluster the lines based on some entity relationship, such as developers working together within the same project. A force directed layout algorithm can also help reduce the number of edge/line crossings. The most popular force directed graph layout algorithms include Fruchterman-Reingold's organic model [11] and Davison-Harel's simulated annealing [8]. Nodes attract and repulse each other based on the force model. Iterative movement minimizes the cumulative force on each node, approaching an equilibrium state and converging to a reasonable layout.

Chapter 3

Design

In this chapter, we describe the design goals, our overall approach and challenges of designing VariantFlow, an interactive storyline visualization technique to visualize textual variations in classical Latin text.

3.1 Design Goals

In designing VariantFlow, our first goal is to support simultaneous graphical representation of the witnesses and lemmas in the base text of a critical edition, revealing their complex relationships. By employing the storyline technique, we aim to depict interaction between witnesses providing variants, the patterns of variants overall and other anomalies in the critical edition above the level of individual lemmas.

Our second goal is to represent the critical apparatus with an intuitive visualization, one that is easily readable and comprehendible to novice readers of Latin texts who do not have sufficient training to read and interpret the apparatus in its canonical form.

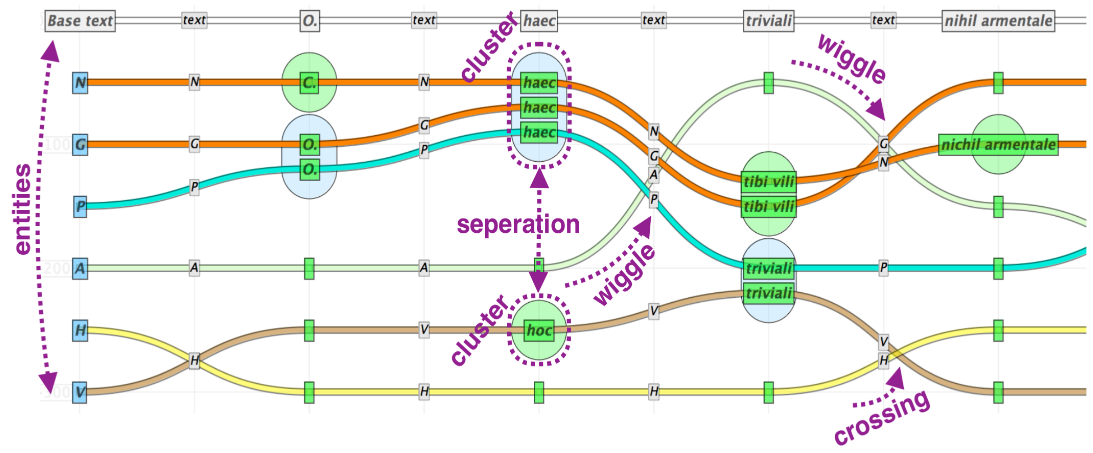


Figure 3.1: Storyline design principals introduced in [23, 28]

Finally, our third goal is to provide a visualization of the apparatus that is responsive to common user interactions in real time.

3.2 Design Approach

Storyline visualizations consist of a time axis and a collection of lines, converging and diverging in the course of their paths, as they progress forward along the time axis. For VariantFlow, we employ the design principals introduced in [23, 28] as follows (see Figure 3.1):

1. Lines represent entities.
2. Lines are clustered based on specific entity relationship.
3. Lines within the same cluster must be adjacent to each other.
4. Clusters should be spaced apart from each other.
5. Line crossings and line wiggles are inevitable, but minimize them as much as possible.

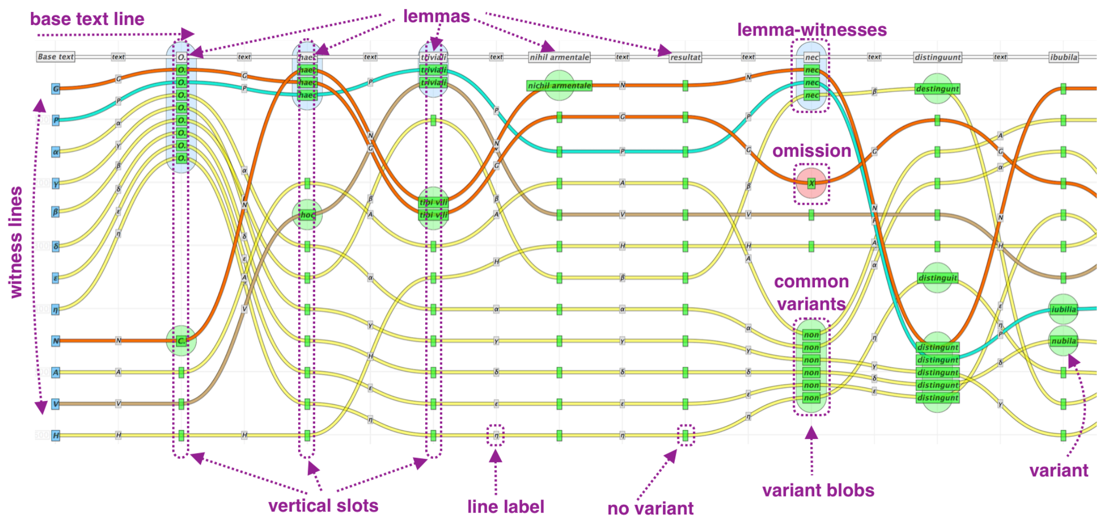


Figure 3.2: Storyline visualization of Giarratano's critical edition of the classical Latin poem *Calpurnius Siculus* [29]. (Elements are described detail in the text.)

6. Balance the amount of empty space in the layout to effectively use screen real estate.

Figure 3.2 shows a storyline visualization of the critical apparatus from a classical Latin poem, *Calpurnius Siculus* [29]. We implemented a prototype of our layout algorithm and visualization by adapting the existing general-purpose graph view in Improvise [31, 14]. In this section, we describe the series of visual encoding decisions we made in crafting VariantFlow.

- **Base text line:** Unlike existing storyline techniques, which show time from left to right, VariantFlow's horizontal axis represents text in reading order. The transparent line along the top (see Figure 3.2) represents lemmas in the base text in a critical edition.

- **Lemmas on base text line:** Lemmas are equally spaced along the top line, from left to right, in reading order. The base text line can be compared to the time axis of the existing storyline visualizations. The lemmas divide the entire view into vertical slots (see Figure 3.2), similar to the time steps or interaction events in existing storyline techniques.

- **Witness line:** Like other storyline techniques, each line represents an entity. In our application case, each line (except the top line) represents a witness (see Figure 3.2).

- **Labels:** Each line is labeled with their witness name. A line label appears between each successive pair of nodes (see Figure 3.2). This provides better traceability and readability of individual lines in the layout.

- **Line color:** Lines of the same color represent witnesses from the same manuscript family (stemma). The boldly colored lines indicate the witnesses from the primary manuscript families. The lightly colored lines represent the witnesses from the inferior (secondary) manuscript families.

- **Variants on witness lines:** With the horizontal axis devoted to text in reading order, the vertical axis is available to represent the relationship between the witnesses and the base text. Within our dataset, the relationships among the witnesses and the base text (lemmas) are determined by their similarities and dissimilarities in text. Each node, representing a variant of a lemma, is vertically aligned with the lemma and horizontally positioned on the contributing witness line. Thus, nodes representing variants of the same lemma are all vertically aligned within that lemma's slot.

- **Common variants in blob:** Multiple witness lines are clustered together in a "blob" or "pack" when they have a common variant (see Figure 3.2). The color of a blob represents its variant category.

  - **green blobs** represent semantic and orthographical variants (simply referred to as *variants*).
  - **blue blobs** represent variants that are the same as the lemma (referred as *lemma-witnesses*). Witnesses that have the same text as a lemma are grouped together in a single blue blob.
  - **red blobs** represent subtractive variants caused by *omission* of a lemma through oversight, erasure, etc. in a witness.

- **Empty boxes:** Empty boxes indicate no textual variation (see Figure 3.2). Like variants, each empty box is vertically aligned with a lemma and horizontally positioned on a witness line, depicting the absence of a variant in a particular witness, for a particular lemma.

In a printed critical edition, scholars have difficulty analyzeing the entire apparatus at once. They can only look at the individual entry or collection of entries in an apparatus on a page. VariantFlow visualization provides an overview of the entire apparatus that helps scholars observe larger patterns and anomalies in variant texts, whereas with printed editions, scholars have no true overview and can only view details at the level of individual lemma. We hypothesize that users may observe the following patterns in variant texts using VariantFlow:

- **Scholarly drift at different scales:** Editors take years to create a critical edition. Over the years, their interest in specific characteristics of the witnesses and their variants, and the editor's knowledge and philological judgement in collating text, may change substantially. Therefore, with VariantFlow, users can look for "drift" in editorial and scholarly decisions, from chapter to chapter or from the beginning of the text to the end in a critical edition.

- **Contamination:** VariantFlow depicts anomalies like contamination, in which a line consistently grouped with one group of witnesses throughout the text suddenly diverges and joins another group, revealing its association with multiple lineages. However, contamination can be difficult to identify in cases in which there are many group changes among the witnesses, caused by a large amount of variation or fragmentation in text.

- **Anomalies in collation decisions:** VariantFlow can be used to identify unusual decisions or conjectures made by the editor during the collation process, like when the editor chooses an outlier variant as a lemma. The outlier could be a variant supported by a single witness (as opposed to the majority of the witnesses) or a variant supported by inferior manuscript families instead of primary manuscript families. Sometimes, editors introduce their own conjecture, ignoring all the witnesses. However, it might be difficult to distinguish these scenarios with VariantFlow, if the editor frequently chooses variants in unconventional ways.

- **Fragmentations:** Fragmentation in a witness is caused by partial destruction of the witness or by the presence of a former manuscript's "waste" or "fragments" within other witnesses. VariantFlow can be used to identify patterns, like lines nicely converged together into few groups at the beginning of a critical edition, that indicate higher quality provenance of the witnesses. However, towards the end of the critical edition, there are few coherent groupings and lines are tangled between groups, revealing many variations and probable fragmentation between witnesses.

3.3 Design Challenges

In this section we describe the design considerations and challenges of crafting storyline visualization specifically as an application to classical Latin text.

Designing an interactive storyline layout for VariantFlow poses the following challenges:

- **Low coherence:** For storyline to be effective, the groupings of lines need to be coherent over time. The challenge with our application case is that grouping of witnesses specifying common variants (see Figure 3.2) does not tend to be coherent across consecutive vertical slots (lemmas in reading order). In contrast, entities in traditional storylines tend to interact with each other over long periods of time, causing the lines to converge across multiple slots in a time line.

- **Spacing and alignments:** In traditional storylines, entities appear and disappear over time, causing the lines to be discontinuous throughout the visualization. Discontinuous lines are easy to position in a suitable place within the visualization or cluster with other lines, as these lines can appear and reappear anywhere throughout the visualization. Discontinuous lines also help avoid line crossings and line wiggles, and the consequent visual clutter. The XKCD storyline uses dotted lines before a character's first appearance and reappearances to imply that their prior locations are unknown. In contrast, lines in a VariantFlow visualization exist throughout the entire text. This makes the spacing and aligning of lines challenging within the storyline layout.

- **Legibility of the layout:** The storyline representation of the critical apparatus needs to be intuitive and easily readable to scholars, as well to students and novice readers of Latin text. Moreover, each storyline element in VariantFlow visualization displays text. In our prototype visualization, we employed the methods introduced in CollateX to position variant texts in nodes [9]. We used in-line labels to indicate witness lines, similar to existing storyline techniques. However, each of these design choices poses usability challenges indicating difficulty in reading the text positioned within the concise space of a node, distinguishing variants with similar spelling, and labels creating visual clutter where lines overlap.

- **Generating suboptimal layout:** Typical design considerations in existing storyline techniques include minimizing line crossing and line wiggling, which are computationally intense problems. Instead of generating an optimal layout with minimal line crossing and line wiggling, we focus on generating a topologically good enough layout, which will allow users to perform various reading and pattern recognition tasks with reasonable speed and accuracy.

- **Best effort responsiveness:** Generating storyline layouts in real time is a substantial challenge to interactive visualization. Existing storyline techniques support limited or no user interaction due to the high cost of producing a layout. An interactive storyline generation technique needs to be able to relayout within 100 ms of an external interaction [26]. A more delayed response makes it harder for typical users to follow the transition of storyline elements across layouts. From user interaction to relayout of storyline includes the steps of dynamic query construction, storyline layout algorithm execution, and rendering storyline elements on the view. Execution of the storyline layout algorithm presents the primary performance bottleneck among these three steps.

Our objective is to ***design a storyline layout algorithm that dynamically generates a legible layout with best effort response to external interaction.***

We are mapping a fairly complex dataset into a storyline visualization compared to those shown in typical narrative storyline techniques. Our design goals and challenges within our chosen application domain calls for design, performance and usability considerations, listed above, that are substantially distinct from existing storyline techniques and their domain areas of application. These considerations led us to design the initial layout algorithm and visualization.

3.4 Limitations

- **Scalability:** We found that the VariantFlow visualization works best with small to medium-sized dataset with no more than 15 lines. Representing the base text as the top line within the same view as the witness lines causes the base text line to disappear during vertical scrolling, causing users to not be able to reference lemmas while reading a variant node far away from the top line. This limits the number of lines we can include in the VariantFlow visualization, causing it to scale poorly. This is an implementation issue rather than a fundamental design issue, and can be corrected in the future designs(by using separate view for the top line, for instance).

- **Wiggle distance and white space:** Apart from minimizing the number of line crossings and line wiggles, state-of-the-art storyline techniques minimize the wiggle distance and empty space to achieve a more balanced and compact layout. However, these design principles and quality metrics call for computing the storyline layout as a combinatory optimization problem. The force model in our prototype FDL algorithm includes a force function that minimizes line crossing and line wiggles. However, we have not yet implemented any force function to explicitly minimize wiggle distance and white space to achieve a more compact layout.

- **Spatial information:** The narrative storyline techniques use color-coded contours as a background on a storyline visualization to show spatial locations where characters interact in a story. These contours occupy relatively fixed positions in the visualization, constraining vertical movement of lines. We excluded the spatial contours from our VariantFlow design because, with our layout algorithm, we are interested first in applying FDL algorithm to the vertical displacement of nodes, keeping their horizontal position unchanged. Constraining the nodes' vertical movement would increase the chances of the FDL algorithm to perform poorly, converging to a suboptimal layout with poor topology. This would affect the interactivity and readability of the layout.

- **Hierarchical relationships:** VariantFlow visualization is not currently designed to represent hierarchical relationships among entities. However, in the future, one might extend packs across multiple vertical slots to encode hierarchical order of entity relationships, such as the spatial contours in narrative storylines [28, 20]. One could also split lines to represent hierarchical relationships among entities. For example, in our application domain, a single witness can provide variants from multiple scribes in chronological order, referred as *witness hands*; meaning, a witness can contain variants from a scribe who made corrections to variants introduced by prior scribes. These variants and the witness hands have hierarchical relationships among them, that can be represented by temporarily splitting a witness line into multiple paths for the variant nodes involving hands. These design perspectives would motivate adding new forces in our existing force model.

Chapter 4

Layout Algorithm

Our VariantFlow layout algorithm was inspired by Ogawa and Ma [23]. In their work, Ogawa and Ma suggested force-directed graph layout algorithm as a potential future research direction for storyline generation. In the following section we describe our approach to generate interactive storylines using force-directed layout algorithm.

4.1 The Overall Approach

As mentioned in the previous chapters, our objective is to ***develop a storyline layout algorithm that dynamically generates legible layout for best effort response to external interaction.***

The existing state-of-the-art storyline techniques compute storyline layouts as combinatory optimization problem with respect to the matrices for legibility and aesthetics of the layout, constrained by their design principals. Although these layout techniques produce optimized layout with minimal line crossing, minimal line wiggles and balanced whitespace, they take seconds to minutes to produce layouts, making them unsuitable for real time user interaction.

Generating storyline layouts in real time is a substantial challenge to interactive visualization. An interactive storyline generation technique needs to be able to relayout within 100 milliseconds of an external interaction [26]. A more delayed response makes it harder for typical users to follow the transition or difference between the storyline elements across layouts. To generate a storyline layout using The Matrix dataset, StoryFlow [20] computes in 0.16 second and Tanahashi and Ma's technique [28] computes in 1.7 second.

This has motivated us to **employ force-directed layout (FDL) algorithm to achieve fast convergence to reasonable layouts, for best effort response to external interaction. We believe that the characteristics of the force-directed layout algorithm will allow for graceful response to a wide variety of interaction types, speeds, and patterns.**

We use a modified version of the Fruchterman-Reingold model [11] for our layout algorithm. Nodes attract and repulse each other. Iterative movement minimizes the cumulative force on each node, approaching an equilibrium state [8]. In our force model, node represents variant of a lemma at a particular point in the text, edges are connections between variants for a given witness, and hyperedge "blobs" are sets of witnesses that have a common variant for a lemma. Application of storylines to critical editions is a special case of graph layout. The variant nodes for each lemma are initially placed in a column (vertical slot), with columns laid out from left to right for each lemma in reading order. All forces and node movements are vertical only.

4.2 Algorithm Overview

Figure 4.1 shows a flowchart of the algorithm for computing the layout of VariantFlow visualization.

Our Layout algorithm consists of two major steps - the organic step and the fixed step.

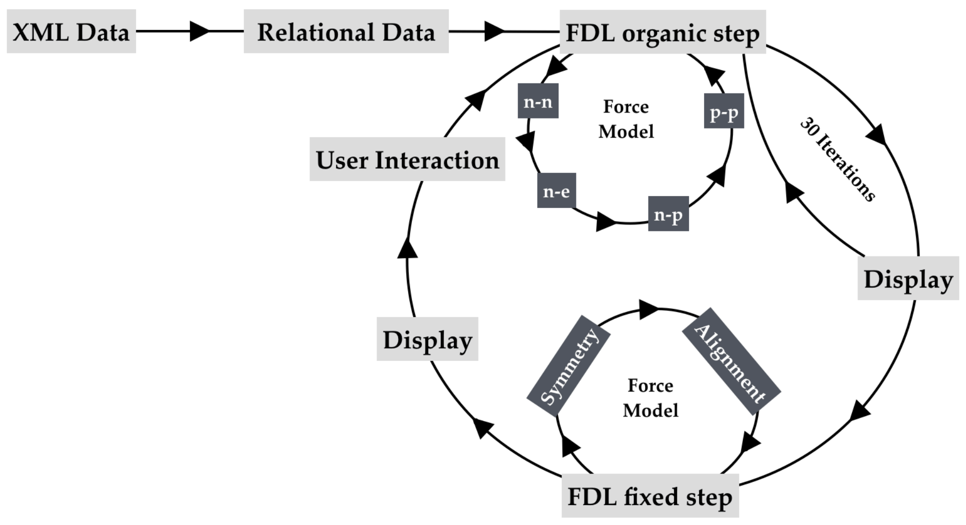


Figure 4.1: Flow chart of our FDL storyline layout algorithm.

4.2.1 Organic Step

In the organic step, we apply a force directed layout algorithm for 30 iterations to converge to a minimum-crossing topological layout. This number was empirically found. Less than 30 iterations usually produce more convoluted layout. More than 30 iterations usually do not make much improvements over the layout achieved by 30 iterations.

4.2.2 Force Model

The force model used in the organic step is as follows:

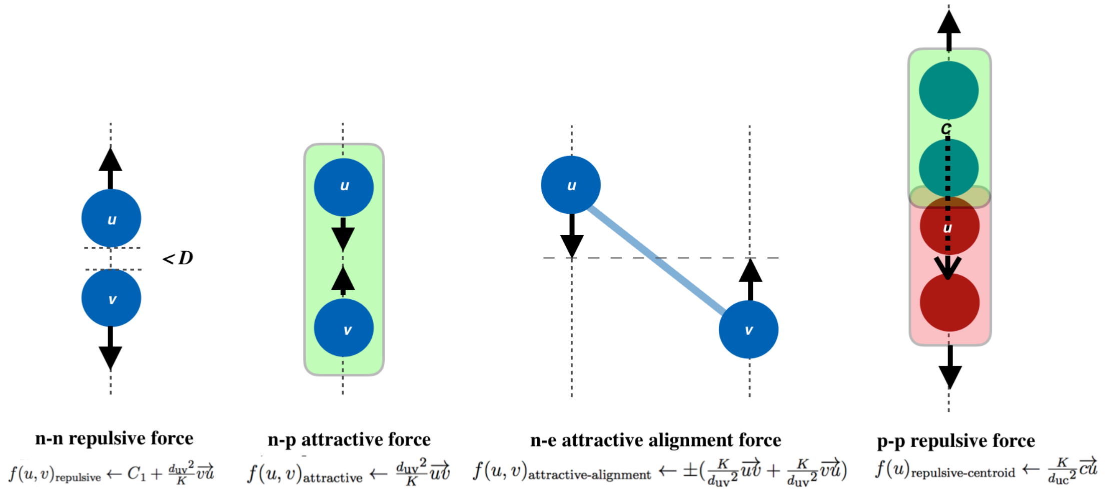

u
v
< D
u
v
u
v
c
u
n-n repulsive force
$f(u,v)_{\text{repulsive}} \leftarrow C_1 + \frac{\text{duv}^2}{K}\overrightarrow{vu}$
n-p attractive force
$f(u,v)_{\text{attractive}} \leftarrow \frac{\text{duv}^2}{K}\overrightarrow{uv}$
n-e attractive alignment force
$f(u,v)_{\text{attractive-alignment}} \leftarrow \pm(\frac{K}{\text{duv}^2}\overrightarrow{uv} + \frac{K}{\text{duv}^2}\overrightarrow{vu})$
p-p repulsive force
$f(u)_{\text{repulsive-centroid}} \leftarrow \frac{K}{\text{duc}^2}\overrightarrow{cu}$

Figure 4.2: Force model used in the FDL organic step of our storyline layout algorithm.

- **Node-node (n-n) forces:** Nodes within the same vertical slot repulse each other, if they are closer to each other than a minimum distance threshold, D. This repulsive force, $f_{\text{repulsive}}$ separate nodes to an equilibrium distance from each other, thus effectively preventing node overlapping within each vertical slot.

- **Node-edge (n-e) forces:** Nodes on consecutive vertical slots, connected by an edge, attract each other vertically, thus pulling each pair of nodes to align horizontally. This force, $f_{\text{attractive-alignment}}$ reduces line crossing and line wiggling.

- **Node-pack (n-p) attractive forces:** Nodes within each variant pack attract each other using a quadratic force centered on equilibrium distance. This force, $f_{\text{attractive}}$ clusters lines within the same pack in each vertical slot.

- **Pack-pack (p-p) repulsive forces:** The packs within the same vertical slot repulse each other, using an inverse-square force, $f_{\text{repulsive-centroid}}$, on their centroids. Overlapping packs are repulsed from each other using a strong repulsive force, $f_{\text{strong-repulsive-centroid}}$. This force separate packs and non-member nodes from other packs within each vertical slot.

4.2.3 Algorithm 1 Description

As shown in Algorithm 1, before applying the force directed algorithm, graph G is initialized with a set of vertical slots, positioned from left to right, according to the order lemmas appear in the input file. Moreover, lines are evenly positioned along x-axis, from top to bottom, according to the order witnesses appear in the input file. Each node is vertically aligned within a lemma's slot and horizontally positioned on a witness line. After the initialization phase, the 30 iterations for the organic step of our force directed layout algorithm begins. During the organic step, the forces described above are applied on each node, on y-direction only, while keeping the horizontal position of the node fixed. Therefore, all forces and distance matrices described in Algorithm 1 are vertical only.

Within each of the 30 iterations, the algorithm loops through each vertical slot in G. If the euclidian distance between a pair of nodes (u, v) within a vertical slot (s) is less than a minimum distance threshold (D), then the none-node repulsive force $(f(u, v)_{\text{repulsive}})$ is applied on u and v to push them further apart.

If a pair of nodes (u, v) within a vertical slot (s) belongs to the same pack (p), the the node-pack attractive force $(f(u, v)_{\text{attractive}})$ is applied on u and v to cluster them together.

If the euclidian distance (d_{uv}) between a pair of nodes (u, v) within consecutive vertical slots (s, s+1), connected through an edge (E), is not zero; meaning, u and v are on the same line, but they are not aligned horizontally, then the node-edge attractive-alignment force ($f(u,v)_{\text{attractive-alignment}}$) is applied on u and v to push them up or down, within their respective vertical slots, in order to make them horizontally aligned.

Packs (P_{s}) within a vertical slot repulse (using $f(u)_{\text{repulsive-centroid}}$) each non-pack-member node (u) from the pack's centroid (c) to make clear separation between the packs and other non-pack nodes. If the euclidian distance (d_{uc}) between a pack's centroid c and a non-pack member node u is less than a minimum distance threshold (D_c), meaning there's an overlap between packs or between a pack and a non-pack-member node (u), then a strong repulsive force ($f_{\text{strong-repulsive-centroid}}$) is applied on u to push it further apart from the pack's centroid c.

We experimented with our force model to achieve faster convergence to a good layout topology. Algorithm 1 includes force functions used in our force model. The constants K, C, C_1, C_2 in the force functions were determined empirically. In the organic step, different forces in our force model compete with each other to reach an equilibrium state. These constants help manipulate the strength of each forces to achieve a good balance. For example, during our experiment, we observed that the n-p attractive force, $f(u,v)_{\text{attractive}}$ usually compete with the n-e attractive alignment force, $f(u,v)_{\text{attractive-alignment}}$. Meaning, if we strengthen $f(u,v)_{\text{attractive-alignment}}$, and weaken $f_{\text{attractive}}$, then our algorithm converges to a layout with less line wiggles and line crossing, however, it fails to effectively cluster lines in packs. On the other, when we strengthen $f(u,v)_{\text{attractive}}$ to cluster lines nicely, our algorithm converges to a layout with more line crossing and line wiggles. Similarly $f(u,v)_{\text{attractive}}$ and $f(u,v)_{\text{repulsive}}$ also compete with each other.

Strengthening one over the other causes our algorithm to generate compact or sparse layout respectively. These constants also have some correlation to the size of G (total number of nodes).

At the end of each iteration in the organic step, graph G is updated with the new node and edge positions calculated after applying these forces on the nodes.

4.2.4 Fixed Step

At the end of 30 iterations with the organic step, we apply strong local symmetry and alignment forces for one more iteration to improve the aesthetics of the storyline layout while keeping the topology of the layout achieved from the organic step. Symmetry forces adjusts all node positions in a pack so that they are very close to each other and distances between adjacent nodes are equal. Symmetric distance between adjacent nodes is quite hard to achieve using the organic forces. Before applying alignment forces, we pre calculate virtual horizontal slot for each line, so that the adjacent lines are equally positioned from each other. After that, we apply strong alignment forces to reposition nodes within each vatical slot to the horizontal slots, while keeping the nodes order within each vatical slot the same. This step reduces unnecessary line wiggles and line crossings, and adds balanced spacing between layout elements, which significant improves the aesthetics of the storyline layout.

We apply these forces iteratively and continuously. In each iteration, we combine the forces to achieve a good balance. This results in convergence to reasonable layouts regardless of earlier layout state. We hypothesize that these characteristics of our model will allow for graceful response to a wide variety of interaction types, speeds, and patterns. This interactive behavior will in turn greatly facilitate the flexible design of coordinated multiple view visualizations that include storyline views.

Term	**Definition**
G(V, E)	Graph with V set of nodes and E set of edges
V	Number of nodes
E	Number of edges
S	Number of vertical slots in G
L	Number of lines in G
P_s	Number of packs in a vertical slot s
D	Minimum Euclidian distance threshold between a pair of nodes
D_c	Minimum Euclidian distance threshold between a pack's centroid and a node that does not belong to that pack
d_{uv}	Euclidian distance between node u and v
d_{uc}	Euclidian distance between node u and a pack's centroid c
$\overrightarrow{uv}$	Unit length vector pointing from node u to v
$\overrightarrow{vu}$	Unit length vector pointing from node v to u
$\overrightarrow{cu}$	Unit length vector pointing from a pack's centroid c to node u
C, C_1, C_2	Some empirical constants
K	$\frac{V}{C}$
iterations	30 is experimentally set as the number of iterations the organic step of the algorithm should run

Table 4.1: Algorithm terms with definitions.

Algorithm 1 FDL Organic Step

```
1: $G \leftarrow initialize(V, E)$
2: {All forces and distance matrices are vertical only}
3: for $k = 1 \rightarrow iterations$ do
4:    for $s = 1 \rightarrow S$ do
5:       $P_{\text{s}} \leftarrow getSlotPacks(s)$
6:       for $i = 1 \rightarrow L$ do
7:          $u \leftarrow getNode(i, s)$
8:          for $j = i + 1 \rightarrow L$ do
9:             $v \leftarrow getNode(j, s)$
10:            $d_{\text{uv}} \leftarrow getEuclideanDist(u, v)$
11:            if $d_{\text{uv}} < D$ then
12:               $f(u,v)_{\text{repulsive}} \leftarrow C_1 + \frac{{d_{\text{uv}}}^2}{K}\overrightarrow{vu}$
13:            end if
14:            for $p = 1 \rightarrow P_{\text{s}}$ do
15:               if $u, v \in p$ then
16:                  $f(u,v)_{\text{attractive}} \leftarrow \frac{{d_{\text{uv}}}^2}{K}\overrightarrow{uv}$
17:               end if
18:            end for
19:         end for
20:         $v \leftarrow getNode(i, s + 1)$
21:         $d_{\text{uv}} \leftarrow getEuclideanDist(u, v)$
22:         if $u, v \in E, d_{\text{uv}} \neq 0$ then
23:            $f(u,v)_{\text{attractive-alignment}} \leftarrow \pm(\frac{K}{{d_{\text{uv}}}^2}\overrightarrow{uv} + \frac{K}{{d_{\text{uv}}}^2}\overrightarrow{vu})$
24:         end if
25:         for $p = 1 \rightarrow P_{\text{s}}$ do
26:            $c \leftarrow getCentroid(p)$
27:            if $u \notin p$ then
28:               $d_{\text{uc}} \leftarrow getEuclideanDist(u, c)$
29:               if $d_{\text{uc}} < D_{\text{c}}$ then
30:                  $f(u)_{\text{strong-repulsive-centroid}} \leftarrow C_2 + K{d_{\text{uc}}}^2\overrightarrow{cu}$
31:               else
32:                  $f(u)_{\text{repulsive-centroid}} \leftarrow \frac{K}{{d_{\text{uc}}}^2}\overrightarrow{cu}$
33:               end if
34:            end if
35:         end for
36:      end for
37:   end for
38:   $G \leftarrow update(V, E)$
39: end for
```

4.3 Time Complexity

Given S being the number of vertical slots and L being the number of lines in graph G, the time complexity of our layout algorithm is $O(SL^3)$.

The order of growth of the running time of Algorithm 1 is

$$\begin{aligned}
&= \sum_{k=1}^{30}\sum_{s=1}^{S}\sum_{i=1}^{L}\left(\sum_{j=1}^{L} c + \sum_{j=1}^{L}\sum_{p=1}^{P_\mathrm{s}} c + \sum_{p=1}^{P_\mathrm{s}} c\right), c = constant, 0 \leqslant P_\mathrm{s} \leqslant L \\
&= 30 * c * S * L(L + LP_\mathrm{s} + P_\mathrm{s}) \\
&= 30 * c * S(L^2 + L^2P_\mathrm{s} + LP_\mathrm{s}) \\
&= 30 * c * S(L^2 + L^2 * L + L * L) \\
&= 30 * c(SL^2 + SL^3 + SL^2) \\
&= 30 * c(2SL^2 + SL^3)
\end{aligned} \tag{4.1}$$

Therefore, the time complexity of our VariantFlow layout algorithm is $O(SL^3)$.

Ogawa and Ma's storyline layout algorithm computes in $O(CT)$ for number of entities C and number of time steps T [23].

The state-of-the-art method proposed by Tanahashi and Ma uses a genetic algorithm (GA) to generate storyline layouts. GA algorithms are computationally expensive. For number of vertical slots S and number of interaction sessions I, the GA step in Tanahashi and Ma's technique computes in $O(S^I)$, which grows polynomially in the number of slots and exponentially in the number of interaction sessions. In addition, computing the layout for each genome takes time complexity $O(CI + STI)$ [28]. Here, C is the number of lines, I is the number of sessions, S is the number of slots, and T is the number of time frames.

On the other hand, the time complexity of StoryFlow is $O({n_\mathrm{e}}^2T + T^3)$ where n_e is the number of entities and T is the number of time frames in a StoryFlow layout [20].

Therefore, the computational complexity of our layout algorithm, $O(SL^3)$ is better than the time complexity of StoryFlow and Tanahashi and Ma's technique.

4.4 Limitations

- **Poor local minima:** FDL algorithms tend to converge to local minima, which produces suboptimal layout with more line crossings and line wiggles and poor aesthetics.

- **Legibility and aesthetics:** For the legibility and aesthetics of the layout, the distance between intra-cluster nodes or lines needs to be symmetric, which an FDL algorithm cannot guarantee.

- **Fixed step limits interactivity:** While FDL algorithms converge to a good layout topology faster, they also usually result in poor layout aesthetics. To overcome this issue, we apply strong local symmetry and alignment forces on nodes in the last iteration of the algorithm (fixed step). This fixed step keeps the topology of the layout achieved from the organic step and simply improves the aesthetics of that layout by applying those strong forces. Although the fixed step significantly improves the legibility and aesthetics of the layout, it interrupts the force system of the FDL algorithm, preventing it from approaching an equilibrium state iteratively. Thus the layout becomes unresponsive to user interactions like dragging or moving nodes, packs or edges.

- **Sensitive to initial position:** Like many FDL algorithm, the initial position of nodes significantly affects the layout convergence of our VariantFlow algorithm. Our FDL algorithm is designed to be applied only along vertical direction of nodes while horizontal position of nodes remain fixed [23]. This requires to initialize our layout with vertical slots for lemmas in reading order, with each variant node vertically aligned with it's lemma's slot and horizontally aligned with it's witness line. We found out that if we initialize the nodes on lines evenly positioned along x-axis, the algorithm converges to a good layout topology within the first 30 iterations. The vertical slots and lines are initialized according to the order lemmas and witnesses appear in the data source. Therefore, the chronological order of lemmas and witnesses in the input file influences VariantFlow layout convergence.

- **Layout complexity:** Due to the FDL algorithm's tendency to settle into a local minimum, the VariantFlow converges to a suboptimal layout. Meaning, our VariantFlow layout can have more line wiggles and line crossing than an optimal layout (with minimum line crossings and line wiggles). Therefore, our layout algorithm tends to generate somewhat more complex layout compared to the existing techniques.

4.5 Implementation

We implemented our layout algorithm in Java. We used the existing general-purpose graph view and a modified version of the existing generic force-directed layout algorithm in Improvise to implement the VariantFlow prototype visualization and layout algorithm [31, 14]. The following section describes the data model and data transformation pipeline for VariantFlow.

4.5.1 Data Transformation

Like many digital humanities project, DLL follows Text Encoding Initiative (TEI) guidelines to represent critical editions in digital form (using XML markups). TEI is a consortium that develops and maintains standards for digital representation of texts. Since 1994, TEI guidelines have been widely used by scholars, editors, publishers, libraries and museums to encode texts for humanities research, teaching, and preservation [2].

The raw input data for our VariantFlow visualization comes from TEI encoded XML file containing the list of apparatus entries for lemmas in the critical edition of the classical Latin poem *Calpurnius Siculus* [29].

The first step in our data transformation pipeline is to map the data from the a TEI-encoded XML file to our data model. We use a basic relational data model for VariantFlow. The data model is consists of four relational tables as follows.

witness *⟨witness_id, name⟩*

lemma *⟨lemma_id, name, poemNo, pageNo, lineNo⟩*

lemmaWitness *⟨lemmaWitness_id, location, name, witness_id⟩*

variant *⟨variant_id, name, lemma_id, witness_id, type⟩*

These tables list individual *witnesses*, *lemmas* and *variants*. A forth $lemmaWitness$ table contains, for each lemma, a list of variants that are the same as the lemma (referred as $lemma - witnesses$). We encode the relationship between each variant and it's lemma using a *lemma_id* as a foreign key. Similarly, the relationship between the $lemmaWitness$ and the *witness* is encoded using the *witness_id* as a foreign key. Moreover, the variant categories (variant, omission, etc.) are encoded using the *type* attribute in the variant table. The *lemma* and $lemmaWitness$ tables contain the page location and the poem or chapter number of each lemma.

The data model could be extended by introducing apparatus types (e.g., note, conjecture) and their attributes, or by introducing additional tables, (for instance to represent hierarchical relationship between witness *hands*).

The next step in our data transformation pipeline involves computing the storyline (graph) elements from the data stored in our data model tables. The graph view consists of three primitive elements: nodes, edges, and packs. First, the data projection phase aggregates occurrences and co-occurrences of the data values in each data dimension and transforms these into unique tuples. Then the graph definition phase collects these tuples into tables that are then mapped into nodes, edges, and packs [15, 32, 14]. For VariantFlow, each unique $lemma - witness - variant$ tuple and $lemma - lemmaWitness$ tuple is mapped into a node. Note that, during these phases, unique $lemma - witness - emptyVariant$ tuples are computed and mapped into empty nodes. Edge and pack computations are more complex. Each set of common witness vs. source-and-destination-node pairs are mapped into an edge. Sets of common $lemma - witness - variant$ tuples and $lemma - lemmaWitness$ tuples are mapped into packs.

The final step in our data transformation pipeline renders and layout the storyline (graph) elements. This step takes the graph primitives generated in the previous step as input. The subsequent $Filtering, Brushing$, and $Layout$ transformations support interaction with nodes, edges, and packs in the view during the automatic VariantFlow layout [15, 32].

4.5.2 Time Complexity

As the general-purpose graph view in Improvise is not tailored for storyline layouts. Thus, the size of the VariantFlow layout, determined by number of lemmas, witnesses and variants, significantly affects the computation time of generating the layout. In generating a VariantFlow layout G with number of nodes n, the organic step computes in $O(n^2 + nP)$ time, which dominates the overall time complexity of the layout generation.

Besides, there are cost associated with computing node's membership in vertical slots S and packs P from each node's positions in the graph view between each iteration. For number of lemmas (vertical slots) S and number of witnesses (lines) L, total number of nodes $n = SL$, which is much larger even for a small size graph. Therefore, the computational complexity of our implementation is poorer than the actual time complexity of our algorithm $O(SL^3)$.

It takes under 2 seconds to generate the layout in Figure 3.2 (not entirely visible) for a storyline with 13 lines, 37 vertical slots and 494 nodes, using a MacBook Pro with an Intel Core i7 processor and 8 GB memory. Most of this time is due to rendering the graph after each iteration, rather than waiting until convergence to render once.

By comparison, to generate a storyline layout using *The Matrix* dataset with 14 lines and 42 time frames, StoryFlow [20] computes in 0.16 second and Tanahashi and Ma's technique [28] computes in 1.7 seconds using a MacBook Pro with Intel Core i7 processor and 4 GB memory.

Existing storyline techniques generate layouts in batch processes and hence do not accommodate interaction well. In contrast, our layout algorithm runs smoothly and continuously in response to interaction, and hence also seems faster (while actually being slower) than existing techniques. These characteristics allow for graceful response to diverse forms of coordinated interaction.

4.5.3 Limitations

- **Design:** The general-purpose graph view in Improvise is not tailored for storyline layouts. Besides increasing the computation complexity of the layout generation, this makes interactions like - selecting or highlighting lines, which is basically witness paths (collection of edges), hard to implement.

- **Interactivity:** Applying strong symmetry and alignment forces on nodes in the last iteration of the algorithm (fixed step) cause the layout to be unresponsive to user interactions. In future, we can replace the fixed step with some more incremental steps which is more suitable to user interaction.

- **Performance:** Our implementation of the algorithm runs slower (takes under 2 seconds) than the performance suitable for real time interaction (100 ms of an external interaction [26]).

- **Scalability:** VariantFlow visualization works best with small to medium-sized dataset with no more than 15 lines. Representing the base text as the top line within the same view as the witness lines causes the base text line to disappear with vertical scrolling, thus causing the users to not be able to reference lemmas while reading a variant node further away from the top line. This limits the number of lines we can include in the VariantFlow visualization, causing it to scale poorly.

Chapter 5

Usability Evaluation

We conducted a preliminary user study to evaluate the legibility and aesthetics of our storyline visualization. In this study we are testing the following four hypotheses:

H1 *The storyline visualization of the critical apparatus is easily readable to students and novice readers.*

H2 *The storyline visualization demonstrates clear separation and groupings of entities based on entity relationships.*

H3 *The storyline algorithm generates topologically good layout.*

H4 *The density and complexity of the layout affects readability of the storyline visualization.*

5.1 Procedure

Nineteen students from a variety of majors participated in this study. To recruit participants, we sent word-of-mouth invitation and email announcements through a campus mailing list. We conducted a separate study session for each participant. The study lasted for an hour and consisted of two sessions with a five-minute break in between. During the first session, participants performed a set of quantitative tasks involving various reading activities using the storyline layout. In the second session, participants answered a set of qualitative questions regarding the legibility and aesthetics of the layout. The study began with the participant signing the consent form and completing a background survey. After that, we provided a brief introduction to our storyline visualization and demonstrated various reading task using the layout. To minimize the influence of learning, we asked participants to perform various training tasks similar to the ones they would be performing in the study. We also encouraged them to ask questions throughout the study. Once the participant was familiar with our storyline visualization, we provided a questionnaire with 18 quantitative tasks.

We used two different storyline layouts in this study. One of the layouts was denser (with 50 variants and 13 lines) than the other (with 25 variants and 7 lines). Half of the participants were presented with the denser layout and the other half were presented with the sparse layout. The study was conducted in an isolated room with the participant seated in front of an Apple MacBookPro with a 15" screen displaying a storyline layout. Participants used typical interaction techniques like pan and scroll using mouse and keyboard.

After completing the quantitative tasks, each participant took a five minute break. After the break the second session of the study began. In this session the participant provided feedback to 7 qualitative questions. All questions required choosing an answer from multiple choices or supplying a brief numeric or text response. Given the ground truth, we compute the error in participant's answer to each quantitative question as zero or one. Participants also provided their confidence level for performing each of the tasks using a 5-step Likert scale.

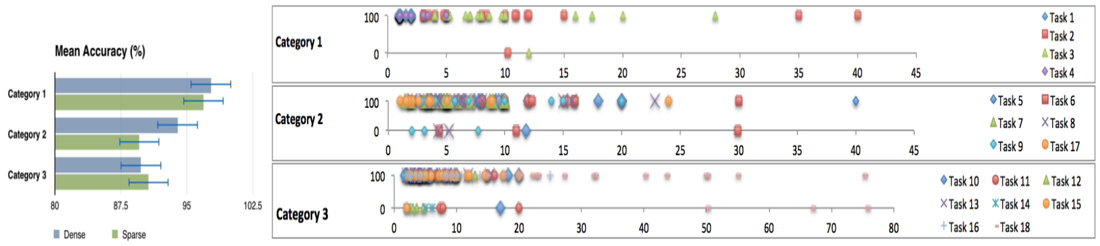


Figure 5.1: Left: Mean accuracy for 3 categories of quantitative tasks with the dense and sparse storyline layouts. Right: User accuracy versus time taken to perform visual metaphor interpretation (category 1), reading (category 2), and pattern recognition (category 3) tasks.

We used a stopwatch to measure the participant's response time and recorded responses using textual transcription and audio recording.

5.2 Task and Data Analysis

Three analyses are of primary interest. First, whether the visual metaphors used in storyline to represent various entities are intuitive. Second, whether the various entity relationships are easily readable using the layout. And finally, whether it is possible to find patterns or similarities between entities using the storyline layout. Therefore, we divided the quantitative tasks into three categories: *visual metaphor interpretation*, *reading*, and *pattern recognition* tasks.

Visual metaphor interpretation (category 1) tasks required participants to identify lines and nodes representing witnesses and lemmas. Participants demonstrated very high mean accuracy (Figure 5.1(left)), confidence, and speed in performing these tasks using both dense and sparse layout. Figure 5.1(right) shows most participants completed these tasks within 20 seconds with 100% accuracy. We found strong positive correlation between accuracy and confidence, and moderate negative correlation between speed and confidence for these tasks.

Reading (category 2) tasks required participants to identify variants for lemmas, categories of variants and witnesses contributing those variants. Participants showed moderately high mean accuracy (Figure 5.1(left)), confidence, and speed to perform these tasks, with both dense and sparse layout. Figure 5.1(right) shows most participants completed these tasks within first 20 seconds with 100% accuracy. A few participants demonstrated poor accuracy and confidence in performing reading tasks that required distinguishing variants with similar spellings. Moreover, we observed that poor performance was often influenced by the small font size or a participant's lack of attention while reading such variants. We also found strong negative correlation between speed and confidence, for these tasks.

Pattern recognition (category 3) tasks required participants to find groupings of lines in blobs, indicating common variants between witnesses. Participants demonstrated good mean accuracy (Figure 5.1(left)), confidence, and speed in performing these tasks with both dense and sparse layouts. As shown in Figure 5.1(right), all tasks (except task 18) were completed within first 20 seconds. Task 18 required participants to trace a set of lines converging at various points through the layout, indicating similarity in how witnesses provide variants. Due to the higher complexity of this task, participants took more time completing it. Moreover, we found strong to moderate negative correlation between speed and accuracy, and between speed and confidence, and also moderate positive correlation between accuracy and confidence for these tasks.

We performed Wilcoxon Rank-Sum test on two unpaired samples, using performance on dense and sparse layouts as the metric for each of the three categories of quantitative tasks. All test results showed p values much larger than 0.05, indicating that the distribution of accuracy, confidence and speed is identical for both of these layouts.

In order to analyze the responses to the qualitative questions, we divided the questions into two categories. Category-1 qualitative questions were related to the aesthetics, spacing, density, and complexity of the storyline layout. All participants agreed that the storyline layout was appealing to them with an average rank of 4.2 (1=least appealing, 5=most appealing). Participants also provided generally positive feedback such as how they really liked how the lines were grouped, that the lemmas and the variants were nicely spaced, and that overall the layout was clear and easy to read. A few participants who used the dense layout stated that some part of the layout had many line crossings, making it difficult to read the line labels. Overall, participants gave an average rank of 3.7 for the spacing, density, and complexity of the layout (1=low, 5=high). Category-2 qualitative questions asked participants to identify the quantitative tasks that were relatively easy or difficult to perform using the storyline layout. Many participants stated that identifying the original text line (transparent) and omissions of a lemma (red blobs) were the easiest. Many participants found tracing groups of lines throughout the layout was relatively difficult, especially in places where there were many line crossings and and a lot of line wiggling.

5.3 Findings

In this section we relate both quantitative and qualitative findings of the study to our hypotheses, and draw conclusions.

Quantitative analysis reveals high mean performance for the reading and visual metaphor interpretation tasks. Most participants completed these tasks within 20 seconds with very high accuracy. A few participants had difficulty distinguishing variants with very similar spellings due to the smaller font size used to display the text. Participants with higher accuracy performed these reading tasks faster, with higher confidence. None of the participants had prior experience with Latin. Therefore, we can accept H1 and conclude that our storyline representation of the critical apparatus is easily readable to novice readers.

Pattern recognition tasks were designed particularly to evaluate H2. Quantitative analysis shows good mean performance for these tasks. A few participants reported that line crossing and line wiggling caused difficulty in visually following groups of lines traveling together throughout the layout. Most of these tasks were completed within the 20 seconds with very high accuracy, which is consistent with all 3 categories of tasks. Participants with higher accuracy performed these reading tasks faster, with higher confidence. However, participants took a longer time to accurately perform tasks with higher complexity. Therefore, we claim that H2 holds true and conclude that our storyline visualization demonstrates clear separation and groupings of entities based on entity relationships.

H3 claims that our storyline algorithm generates topologically good layouts. Meaning, given prioritization of the witnesses, there exists an optimal layout with a minimal number of line crossings and line wiggles. Our FDL algorithm produces a layout that has more line crossings and line wiggles than the optimal one. However, our quantitative analysis indicates participants were able to perform various reading and pattern recognition tasks with high mean accuracy, confidence, and speed. Our qualitative analysis demonstrates participants found the VariantFlow layout to be aesthetically pleasing and nicely spaced with moderate density and complexity. Therefore, we can accept H3 and conclude that even though our layout does not meet the standard of the optimized layout, it is a sufficient layout for readability.

To evaluate H4, we ran a Wilcoxon Rank-Sum test on performance data drawn from two samples (dense and sparse) for each of the three categories of quantitative tasks. The results indicate no significant difference in accuracy, confidence or speed for tasks performed either with dense or sparse layout. Therefore, we reject H4 and claim that density and complexity of the layout does not affect the readability of our storyline visualization.

The results from our preliminary user study discussed in this section demonstrates that our storyline visualization of the critical apparatus was easy to use for novice readers, who performed various reading and pattern recognition tasks with accuracy and efficiency that was not significantly affected by the complexity of the layout.

Chapter 6

Discussion and Future Work

In this chapter, we provide a brief usability and utility analysis of the VariantFlow visualization based on our objectives and evaluation outlined in Chapter 3 and Chapter 5, respectively.

6.1 Analysis

Our first goal is to help scholars observe larger patterns and anomalies in variant texts—including "scholarly drifts", fragmentations in witnesses, contamination, unconventional collation decisions, and conjectures—using the VariantFlow visualization of a critical apparatus. To support these visual pattern recognition tasks effectively, VariantFlow employs a storyline technique to trace the witnesses that have many variants in common throughout the text, revealing their similarities and dissimilarities. The results from our usability evaluation, discussed in Chapter 5, demonstrate that VariantFlow visualization helps users to perform several pattern recognition tasks efficiently.

Our second goal is to help students, who do not have sufficient training to read or interpret the apparatus in a critical edition, to read and comprehend the information in the apparatus using VariantFlow. The results from our usability evaluation shows that users with no experience with Latin were easily able to read and interpret information; they could lookup different variants of a lemma, identify a variant category from the color of a pack, and identify the list of witnesses providing the same variant as a lemma. Users also found the visual metaphors used in VariantFlow quite easy to use. While a few users experienced difficulty in simply reading the text due to small font size, most users performed reading and visual metaphor interpretation tasks efficiently.

Finally, our third goal is to develop a storyline layout algorithm that dynamically generates legible layout for best effort response to external interaction. The aesthetics and legibility of a layout is usually determined by the number of line crossings and line wiggles and the balanced use of whitespace in the layout. Minimizing line crossings and line wiggles is a computationally intense problem. With our FDL algorithm, we attempted to reduce line crossings and line wiggles by applying node-edge attractive forces on pair of nodes connected by an edge. Using the force model described in Chapter 4, our FDL algorithm tries to converge to an equilibrium state (typically an improved best suboptimal layout) by clustering lines based on entity relationships, while untangling line crossings and straightening lines in each iteration. In the future, we would like to explore alternative force functions and fine tune our force model to achieve an even better layout.

The results from our usability evaluation show that the complexity of the layout did not affect the accuracy and efficiency of users performing several reading and pattern recognition tasks. Although a few users complained about visual clutter in parts of the layout where there were more line crossings, most of them performed reading and pattern recognition tasks with high speed and accuracy. Users also found the VariantFlow layout to be aesthetically pleasing and nicely spaced with moderate density and complexity. Therefore, we conclude that even though our layout is a suboptimal layout—meaning there are more line crossings and line wiggles in our layout than in an optimal layout (with minimum line crossings and line wiggles), it is sufficient for readability. Our current implementation of the algorithm takes less than 2 seconds to generate a layout with 13 lines, 37 vertical slots and 494 nodes, which is a little slower than the response time suitable for external interaction. Most of this time is due to rendering the storyline layout after each iteration. Our layout algorithm runs smoothly and continuously in response to interaction, and hence also seems faster (while actually being slower). With a smaller dataset, our algorithm converges to a layout that is sufficient for readability within 100 ms, a time that is perceptually suitable for immediate iteration. Therefore, we conclude that our VariantFlow algorithm dynamically generates legible layout for best effort response to external interaction.

6.2 Generalization

Although our VariantFlow layout algorithm was applied to visualize variant texts in Latin poems, it can easily be adapted to other application domains, such as narratives in movies or the software evaluation storylines in a software development project's version control system dataset. Like existing storyline techniques, VariantFlow can effectively represent entities like movie characters or software developers as lines. Instead of representing lemmas in reading order, VariantFlow can represent time along the horizontal axis, over which the entities interact with each other. Lines in existing storyline techniques progress through time (along the horizontal axis), from left to right, converging and diverging in the course of their path, depending on entity relationships. In storylines of movie narrative, the lines of character bundle together for the duration(s) of time that the characters interact with each other. Typically, character lines group together based on the geographic proximity of the characters in the movie over time. Similarly, software evaluation storylines cluster lines based on the developers who work on the same project over time [23]. As the developers stop making code commits for a project and move to other projects, the lines representing those developers diverge from the previous cluster and converge to a new cluster. Here, the lines are grouped based on the developers' similarities in sharing software projects. Like representing similarity in variant texts, VariantFlow can also effectively represent other types of similarity matrices, such as the ones discussed above. Therefore, we believe the VariantFlow layout algorithm and visual representation generalize well.

6.3 Limitations and Future Work

This thesis work is part of ongoing research. Here, we presented our prototype visualization and layout algorithm followed by a usability evaluation to verify the viability of the research direction. We observed positive results and received positive feedbacks in our usability evaluation. This has motivated us to start planning revision and extension of the storyline design and implementation as follows:

- In computing storyline layouts with an FDL algorithm, the search space contains several local minima. In the future, we would like to add random restarts to our algorithm to escape from local minima and broaden the range of search.

- FDL algorithms tend to converge in local minima. For the legibility and aesthetics of the layout, we wanted to keep the distance between nodes within a pack symmetric, which a FDL algorithm cannot guarantee. To overcome this issue, we apply strong local symmetry and alignment forces on the nodes at the last iteration of the algorithm (the fixed step). This fixed step adopts the *topology* of the final layout achieved from the organic step and simply improves the *aesthetics* of that layout. Although the fixed step significantly improves the legibility and aesthetics of the layout, it comes to a price. The fixed step interrupts the force system of the FDL algorithm, causing it to stop approaching an equilibrium state iteratively. Thus the layout becomes less interactive to user interactions, such as dragging nodes, packs, or edges. We would like to replace the fixed step with a more incremental adjustment of the layout aesthetics that is more suitable to user interaction.

- The drastic change in the layout aesthetics between the organic step and the fixed step can make it difficult for users to follow the movement of storyline elements across these two steps. Therefore, we have considered implementing *tweening* animation, in more incremental and aesthetic adjustment steps, to make the movement of the layout elements easy to follow, while preserving the user's mental model.

- In designing VariantFlow, representing the base text as the top line within the same view as the witness lines causes the base text line to disappear with vertical scrolling, making it harder for users to refer to lemmas while reading variant nodes far away from the top line. This limits the number of lines we can include in the VariantFlow visualization, causing it to scale poorly in the number of witnesses. To overcome this issue, we would like to use a separate view for the lemmas, with horizontal scrolling synchronized but vertical scrolling independent, so that the lemmas are always visible.

- VariantFlow visualization is not currently designed to represent hierarchical relationships among entities. However, in the future, we can use packs (or design new hyperedge blobs) to extend across multiple vertical slots to encode hierarchical order of entity relationships, like the spatial contours in narrative storylines [28, 20]. We can also split lines to represent hierarchical relationships among entities. For example, in our application domain, a single witness can provide variants from multiple scribes in chronological order, referred to as *witness hands*; meaning, one scribe can make a correction to a variant introduced by a previous scribe. Therefore, these variants and the witness hands have a hierarchical relationship among themselves which can be represented by splitting the witness line around the variant nodes involving hands. These design perspectives would motivate adding new forces in our existing force model to handle splits in lines.

- The next step is to implement a complete visualization tool in Improvise for the DLL desktop app, incorporating our VariantFlow technique. The tool would consist of a VariantFlow view, a text view, and brushed linking between the text and the VariantFlow elements. The tool might also contain collections of switches to manipulate the VariantFlow layout content, like filtering witnesses or lemmas, showing or hiding packs, showing or hiding empty boxes, etc.

- We are particularly interested in implementing the following interactions that affect the topology of the VariantFlow layout:
  - **Witness/lemma spacing:** Users can interactively change the default spacing between lines and lemmas.
  - **Horizontal/vertical force direction constant:** Users can dynamically change the constants used in the force model to determine the topology of the layout.
  - **Filtering lemmas/horizontal scrunching** Extend our algorithm to apply horizontal forces on nodes within each vertical slot to smoothly transition horizontally into a denser layout while keeping the vertical positions unchanged.

The study of literature is changing dramatically by incorporating new opportunities the digital technology presents. Literary scholars, editors, students and teachers are increasingly incorporating data visualization in their research and academic work. Studying critical editions and their apparatus is of great interest to literary scolders and editors. However, they are only able to look at individual entry or collection of entries in an apparatus on a page, instead of the entire text. Moreover, students and novice readers of Latin texts are typically unable to read or interpret the critical apparatus in it's canonical form.

This thesis presents VariantFlow, an interactive storyline visualization of the critical apparatus that is easily readable and comprehendible to the students, teacher and novice readers of Latin texts, who do not have sufficient training to read the apparatus in it's canonical form. VariantFlow augments scholarly and editorial work by providing an overview of the critical apparatus, to observe larger patterns and anomalies in variant texts. This thesis contributes a storyline layout algorithm for VariantFlow visualization, that dynamically generates legible layout for best effort response to external interaction. We belive that the characteristics of our force-directed layout algorithm will allow for graceful response to a wide variety of interaction types, speeds, and patterns.

In this thesis, we have implemented a prototype of our layout algorithm and visualization. We also conducted a preliminary usability evaluation to validate the legibility and effectiveness of our VariantFlow visualization.

Chapter 7

Conclusion

This thesis work presents outcomes of an initial phase in ongoing research. Before moving to the next phase, we sought to investigate the legibility and effectiveness of our VariantFlow (storyline) layout. The next phase of the research includes implementation of user interactions to dynamically generate the storyline layout in interactive time, followed by an extensive usability evaluation of various interaction techniques with storyline visualization.

We anticipate that the dynamic interactive querying capability of our VariantFlow visualization will help users explore and analyze relational structures in ordered datasets. The high dynamics of interactive querying requires layout techniques that will dynamically generate decent layouts in real time. Current state-of-the-art storyline techniques do not produce layouts in real time, making them generally unsuitable for use with many common kinds of interactive querying like dynamic filtering. The work in this thesis accomplishes several key steps towards this goal.

Bibliography

[1] Juxta. http://www. juxtasoftware.org/.

[2] TEI: Text Encoding Initiative. http://www.tei-c.org/.

[3] T. L. Andrews. Prolegomena to a Critical Edition of the Chronicle of Matthew of Edessa, with a Discussion of Computer-Aided Methods Used to Edit the Text. In *Oxford: University of Oxford*. http://ora.ouls.ox.ac.uk/objects/uuid(accessed 12 June 2013) 2009.

[4] Tara L. Andrews and Caroline Macé. Beyond the tree of texts: Building an empirical model of scribal variation through graph analysis of texts and stemmata. In *Literary and Linguistic Computing*, volume 28, pages 504–521, December 2013.

[5] S. Boodts. A New Critical Edition of Augustine's Sermo 170. With a Tentative Analysis of the Stemmatic Position of the De Lapsu Mundi Collection. pages 50: 185–225. Sacris Erudiri: a Journal on the Inheritance of Early and Medieval Christianity, 2012.

[6] L. Byron and M. Wattenberg. Stacked Graphs - Geometry & Aesthetics. In *IEEE Transactions on Visualization and Computer Graphics*, pages 14(6), 1245–1252, 2008.

[7] W. Cui, S. Liu, L. Tan, C. Shi, Y. Song, Z. Gao, H. Qu, and X. Tong. TextFlow: Towards Better Understanding of Evolving Topics in Text. In *IEEE Transactions on Visualization and Computer Graphics*, pages 17(12):2412–2421, 2011.

[8] Ron Davidson and David Harel. Drawing graphs nicely using simulated annealing. In *ACM Transactions on Graphics*, volume 15, pages 301–331, October 1996.

[9] Ronald H. Dekker and Gregor Middell. Computer-Supported Collation with CollateX: Managing Textual Variance in an Environment with Varying Requirements. In *Supporting Digital Humanities*, pages 17–18. University of Copenhagen, Denmark, 2011.

[10] M. Dork, D. Gruen, C. Williamson, and S. Carpendale. A visual backchannel for large-scale events. In *IEEE Transactions on Visualization and Computer Graphics*, pages 16(6):1129–1138, 2010.

[11] Thomas M. J. Fruchterman and Edward M. Reingold. Graph drawing by force-directed placement. In *Software- Practice and Experience*, volume 21, pages 1129–1164, November 1991.

[12] S. Havre, B. Hetzler, and L. Nowell. ThemeRiver: In Search of Trends, Patterns, and Relationships. In *IEEE Transactions on Visualization and Computer Graphics*, pages 8(1):9–20, 2002.

[13] Stefan Janicke, Annette Geßner, Marco Buchler, and Gerik Scheuermann. Visualizations for Text Re-use. In *International Conference on Information Visualization Theory and Applications (IVAPP)*. 59-70, June 2014.

[14] T.J. Jankun-Kelly, Tim Dwyer, Danny Holten, Christophe Hurter, Martin Nöllenburg, Chris Weaver, and Kai Xu. Scalability Considerations for Multivariate Graph Visualization. In Andreas Kerren, Helen C. Purchase, and Matthew O. Ward, editors, *Multivariate Network Visualization*, volume 8380 of *Lecture Notes in Computer Science*, chapter Scalability Considerations for Multivariate Graph Visualization, pages 207–235. Springer, 2014.

[15] T.J. Jankun-Kelly, Tim Dwyer, Danny Holten, Christophe Hurter, Martin Nöllenburg, Chris Weaver, and Kai Xu. Scalability Considerations for Multivariate Graph Visualization. In *Lecture Notes in Computer Science 8380, Multivariate Network Visualization, eds. Andreas Kerren, Helen C. Purchase, and Matthew O. Ward*, pages 207–235, Springer, Berlin, 2014.

[16] E.J. Kenney. The Classical Text: Aspects of Editing in the Age of the Printed Book. In *Sather Classical Lectures*, volume 44. Berkeley: University of California Press, 1974.

[17] N. W. Kim, S. K. Card, and J. Heer. Tracing Genealogical Data with TimeNets. In *Proceedings of the International Conference on Advanced Visual Interfaces*, pages 241–248, 2010.

[18] M. Krstajic, M. Najm-Araghi, F. Mansmann, and D. Keim. Incremental visual text analytics of news story development. In *In Proceedings of Conference on Visualization and Data Analysis, VDA*, 2012.

[19] J. Leskovec, L. Backstrom, , and J. Kleinberg. Meme-tracking and the dynamics of the news cycle. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining, KDD '09*, pages 497–506, New York, NY, USA, 2009. ACM.

[20] Shixia Liu, Yingcai Wu, Enxun Wei, Mengchen Liu, and Yang Liu. StoryFlow: Tracking the Evolution of Stories. In *IEEE Transactions on Visualization and Computer Graphics*, volume 19, pages 2436–2445, December 2013.

[21] Paul Maas. Textual criticism. Oxford, Clarendon Press, 1958.

[22] R. Munroe. XKCD. http://xkcd.com/657/, December 2009.

[23] Michael Ogawa and Kwan-Liu Ma. Software evolution storylines. In *Proceedings of the 5th international symposium on Software visualization*, pages 35–42. ACM, 2010.

[24] K. Reda, C. Tantipathananandh, A. Johnson, J. Leigh, and T. Berger-Wolf. Visualizing the Evolution of Community Structures in Dynamic Social Networks. In *Comp. Graphics Forum*, pages 30(3):1061–1070, 2011.

[25] Desmond Schmidt and Robert Colomb. A data structure for representing multi-version texts online. In *International Journal of Human-Computer Studies*, 67(6), pages 497–514, 2009.

[26] B. Shneiderman. Dynamic queries for visual information seeking. In *IEEE Software*, volume 11, pages 70–77. [Online]. Available: http://dx.doi.org/10.1109/52.329404, November 1994.

[27] Yuzuru Tanahashi, Chien-Hsin Hsueh, and Kwan-Liu Ma. An Efficient Framework for Generating Storyline Visualizations from Streaming Data. In *IEEE Transactions on Visualization and Computer Graphics*, volume 21, pages 730–742, 2015.

[28] Yuzuru Tanahashi and Kwan-Liu Ma. Design Considerations for Optimizing Storyline Visualizations. In *IEEE Transactions on Visualization and Computer Graphics*, volume 18, pages 2679–2688, December 2012.

[29] Titus Calpurnius Siculus and Caesar Giarratano. *Calpurnii Et Nemesiani Bucolica.* 1910.

[30] F. Viegas, M. Wattenberg, and K Dave. Studying Cooperation and Conflict between Authors with History Flow Visualizations. In *Computer Human Interaction,Vienna, Austria*, 2004.

[31] Chris Weaver. A User Interface for Interactive Construction of Highly-Coordinated Visualizations. In *University of Wisconsin-Madison*, June 2006.

[32] Chris Weaver. Multidimensional Data Dissection Using Attribute Relationship Graphs. In *Proceedings of the IEEE Conference on Visual Analytics Science and Technology 2010*, Salt Lake City, UT, October 2010.

[33] M. L. West. Textual Criticism and Editorial Technique: Applicable to Greek and Latin Texts. Walter de Gruyter, 1973.