ShakesPeer: A Tool for Visualizing Character Relationships in Shakespearean Literature

Mint Tanprasert, Frances Sin, and Kevin Chow

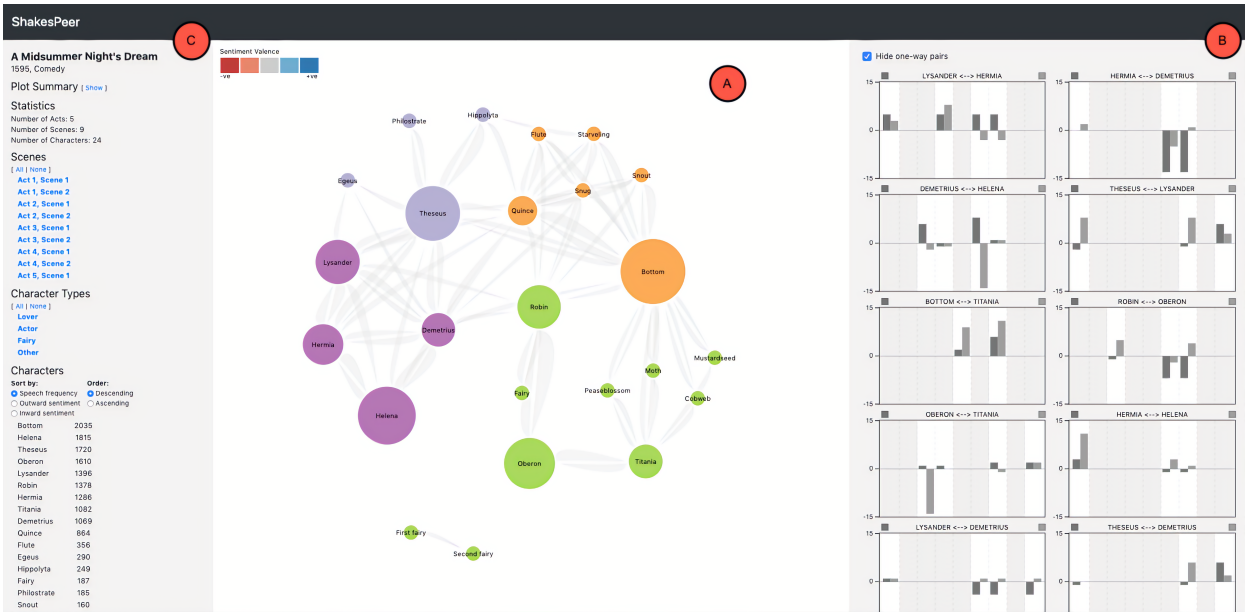

Fig. 1: Overview of *ShakesPeer* for visualizing *A Midsummer Night's Dream* consisting of (a) the *Character Overview*, (b) the *Character-Pair view* and (c) the *Sidebar* menu.

Abstract—*ShakesPeer* is an open-source, interactive tool that visualizes character-to-character relationships in Shakespearean literature. It supports in-depth analysis of sentiment dynamics between characters, allowing users to identify influential moments in the narrative as well as a character's friends and enemies at different points in the story. *ShakesPeer* provides two main views: (1) the *Character Overview*, which spatially maps out the connections between every character in a play, and (2) the *Character-Pair view*, which provides detailed information about the development and flux of sentiment between each pair of characters over the course of the story.

1 Introduction

Character relationships are, in many ways, central to the development of any story. This is particularly true for Shakespearean plays, which often consist of rich, faceted characters with dynamic inter-character relationships. Visualizing these relationships can help readers develop a more nuanced understanding of the story and uncover insights that may not be obvious from reading the text alone.

We propose a new tool, *ShakesPeer* [1], which visualizes relationships between Shakespearean characters. By analyzing the dialogue from a play, we can derive attributes such as character-to-character sentiment and engagement. We hope that *ShakesPeer* will help readers gain a better understanding of the plot, in particular related to the characters, and provide a starting point for deeper literary analysis. For the scope of our work, we focus only on visualizing Shakespeare's famous comedy, *A Midsummer Night's Dream*. However, *ShakesPeer* is not tailored to any specific characteristics of this play and has sufficient scalability to visualize any other plays by Shakespeare as well as plays by other authors of similar format.

In Section 2, we will discuss previously existing work about visualization of Shakespearean texts and sentiment analysis. In Section 3, we will describe our data as well as the data and task abstraction. In Section 4, we will present our visualization solution. Then, in Section 5, we will describe the implementation process as well as the contribution of each participant in the project. In Section 6, we will present our result based on scenario use. Finally, in Section 7, we will discuss the challenge in our design process, the strengths and limitations of the tool, and possible future extensions of the project.

2 Related Work

2.1 Digitized Shakespearean texts

Shakespearean texts have been a popular source for machine analysis and visualization for two reasons. Firstly, its drama script format makes it easy to identify the hierarchical structure of the text. The script also simplifies character-based analysis, since each speech has an explicit character label. Another reason is that the original Shakespearean texts have been thoroughly aggregated, annotated and converted into a digital format, making it a convenient source of data to manipulate and analyze. Some examples of such collections include the Folger Digital Texts [2] project, which provides complete TEI-annotated Shakespearean

• *Mint Tanprasert, Frances Sin, and Kevin Chow are MSc students in Computer Scinece at the University of British Columbia. Their emails are: {tt1996, francsin, kchowk}@cs.ubc.ca.*

ShakesPeer (2019) is a project for CPSC 547: Information Visualization.

[1] https://github.com/kevin-chow/ShakesPeer/

[2] https://www.folger.edu/folger-digital-texts

plays, and Open Source Shakespeare (OSS) [3], which provides statistical features and keywords search on Shakespearean texts.

2.2 Visualization of Shakespearean texts

Shakespeare's play scripts have been visualized in three main ways: text navigation, word-based visualization, and character-based visualization. For text navigation, the related tasks are comparison between different versions of text, such as between different translations [2], and exploration of specific parts of the text, taking into account its positioning in the whole script [3]. Word-based visualization is mostly done in the form of speech distribution visualization. Zakovich orders the speech distribution per scene of the two most important characters in each Shakespeare's story in circles. Other kinds of word-based visualization include text collages, where the size of a word encodes its frequency in the story, and scatterplots of words, showing their correlations and similarity [13]. Character-based visualization are often presented as node-link networks of character co-occurrences in a scene. Grandjean creates such networks for each one of Shakespeare's tragedies, allowing high-level comparison of network structures between stories [3]. Rather than looking at character co-occurrences, *ShakesPeer*'s node-link visualization is based on the conversations between characters. If a pair of characters appear in the same scene but do not speak with each other, there is no link in between them.

2.3 Sentiment Analysis

Sentiment analysis is a text analysis technique, which is widely used to process fictional texts. For Shakespearean texts in particular, Nalisnick et al. performs sentiment analysis on every pair of character in *Hamlet* and *Macbeth* and creates a node-link network of characters where each link is color-coded as green, if the two characters share a "good/positive" relationship, and red for the opposite [9]. Sentiment analysis can also shown progression of the story as a whole, independent of each character's contribution. This technique has been applied to all Tolkien's work [5]. Finally, word emotion classification is very similar to sentiment analysis, albeit with more categories of outputs (specific emotions instead of positive versus negative). Mohammad uses this technique to process *Hamlet* (tragedy) and *As You Like It* (comedy) and shows how emotions change across time for different genres of Shakespearean texts [7]. For our visualization, we closely follow the aforementioned approach of Nalisnick et al. by performing sentiment analysis on each pair of characters who interact in *A Midsummer Night's Dream* [9].

3 Data and Task Abstractions

3.1 Domain Background

We focus on *A Midsummer Night's Dream* as an exemplary piece of Shakespearean literature that our tool can support. *A Midsummer Night's Dream* is a comedy written in 1595 that involves multiple interconnecting plots, weaved together by the wedding celebration of Theseus and Hippolyta. We decided to choose this play because of its popularity and plot, which involves complex, changing relationships between its characters who frequently fall in and out of love with each other. This type of dynamic storyline would best demonstrate the usefulness of *ShakesPeer*.

The raw data consists of the full script of *A Midsummer Night's Dream*, collected from Folger Digital Texts. This play has five acts with nine scenes. There are two scenes per act, except for the last act (Act 5), which only has one scene. In total there are 605 speeches, 2290 lines, and 16,511 words, with an average of 27.29 words per speech. A speech refers to a sequence of words spoken by a character, which can consist of either a single word or multiple lines of words in a soliloquy. There are 23 characters, and 71 pairs of characters who have some speech towards one another. Character pairs can either be one-way or two-way. One-way pairs are when character A talks to character B, but B does not talk to A. Two-way pairs are when both A and B have some speech directed towards the other.

[3] http://www.opensourceshakespeare.org

Table 1: Summary of all Data Attributes.

Attribute Name	Attribute Type	Description
Character name	Categorical	The name of a character.
Character type	Categorical	Label assigned to a character's role, consisting of: lover, actor, fairy, and other.
Scene number	Ordered (ordinal)	Ranges from 1 to 9, corresponding to Act 1, Scene 1 to Act 5, Scene 1.
Sentiment	Ordered (quantitative, diverging)	The summation of sentiment scores (per word, according to the selected lexicon) of all speeches from a speaker to a recipient for a scene or a set of scenes.
Character-to-character engagement	Ordered (quantitative, sequential)	The summation of the total number of words in all speeches from a speaker to a recipient in a scene or a set of scenes
Overall engagement	Ordered (quantitative, sequential)	The summation of the total number of words spoken by a speaker in a or a set of scenes

3.2 Data Abstraction

We will break down *A Midsummer Night's Dream* into two dataset types: (1) a network for character relationships and (2) a table for per-scene information, such as character word counts. All data attributes are also summarized in Table 1 for ease of reference.

3.2.1 Character Relationship Network

In the character relationship network dataset, the nodes are characters, and the links between them encapsulate sentiment and engagement data for each scene. Each node has two attributes: **character name** and **character type**. Each link in the network has a list of data items, one per scene. Each item in the list has five types of attributes: **scene number**, **sentiment** and **engagement** of character A towards character B and the **sentiment** and **engagement** of B towards A (if the link represents a two-way pair).

Character name is categorical with 23 unique levels, each corresponding to one of the characters in the story.

Character type is also categorical, with 4 unique levels, each corresponding to the character's role in the play (lover, fairy, actor, or other). This was manually annotated based on prior knowledge about *A Midsummer Night's Dream*.

Scene number was captured as an ordinal attribute, which ranges from 1 (representing Act 1, Scene 1) to 9 (Act 5, Scene 1).

Sentiment was captured as an ordered, quantitative, but diverging attribute, with a range of -15 to 15, based on the Bing Liu sentiment lexicon [6]. This range is specific to *A Midsummer Night's Dream*, as it is calculated based on speech from that play. Negative numbers are negatively valenced and positive numbers are positively valenced.

Character-to-character engagement represents the total number of words that a character speaks to another. For engagement, direction matters, as the amount of words that A speaks to B will likely be different from B to A. This attribute was captured as an ordered, quantitative, and sequential attribute, and may range from 0 to 16,511 (the total number of words in *A Midsummer Night's Dream*).

3.2.2 Per-Scene Data Table

Information for each scene was captured in a simple flat table, where each row of the table represents a scene in the story. The key of the table is an explicit **scene number** attribute, similar to the one in the character relationship network. The value attributes of the table include the **overall engagement** of each character in that particular scene. Because there are 23 characters in *A Midsummer Night's Dream*, each row has 23 value attributes.

For each character, we calculated their **overall engagement** by simply summing up the number of words of their speech. In contrast to the character relationship network, where the **character-to-character engagement** was directed towards a particular character, **overall engagement** ignores who the speech was directed to. However, **overall engagement** is also ordered, quantitative, and sequential, and shares the same range of values as **character-to-character engagement**.

Table 2: Summary of the What-Why-How analysis of *ShakesPeer*.

System	ShakesPeer
What: Data	Network, flat table
What: Derived	2 ordered key attributes, 1 categorical attributes, 2 quantitative attributes (1 diverging, 1 sequential)
Why: Tasks	Discover, compare, identify, summarize
How: Encode	Node-link graph, Grouped bar charts
How: Manipulate	Select, hover, linked highlighting
How: Facet	Small multiples
Scale	23 characters, 71 character pairs, 9 scenes, 605 speeches, and 16,511 words

3.3 Task Abstraction

ShakesPeer supports four main tasks, which are as follows:

- **T1:** Discover how sentiment and engagement between a pair of characters changes throughout the course of the play.
- **T2:** Compare the relationships between a pair of characters (e.g., A-B and B-A) or across several pairs of characters (e.g., A-B and C-D) in terms of sentiment and engagement.
- **T3:** Identify key points in the story where major changes in relationships occur (in terms of sentiment and/or engagement).
- **T4:** Summarize a scene or a set of scenes by providing an overview of all the characters' relationships towards one another.

4 Solution

In this section, we describe our visualization solution and analyze it using the What-Why-How framework from *Visualization Analysis and Design* [8]. The summary of the analysis is shown in Table 2. *ShakesPeer* consists of two main views: *Character Overview* and *Character-Pair view*. The *Character Overview* is designed to provide a overview of all character relationships in the play or in a scene of the play. The *Character-Pair view* is designed to provide a detailed temporal view into the relationship between pairs of characters with sentiment values and engagement counts between each pair in each scene.

4.1 Character Overview

The *Character Overview* consists of two main components: the node-link graph and the *Sidebar* menu. In the node-link graph, each node represents a character and each directed link represents the existence of engagement between two characters as shown in Figure 3 The default graph gives an overview of engagement and sentiment data from every scene for every character. Although there are only 23 characters in *A Midsummer Night's Dream*, other Shakespearean plays could have up to over 60 characters, which could lead to an explosion of character relationships, and consequently, links in the graph. Therefore, to mitigate the "hairball effect", the *Sidebar* menu was added to allow the user to filter the graph to only look at specific scenes or characters. The user can select a subset of scenes or character types that they would like to include or exclude. Moreover, to assist the user in quickly finding interesting information, a list of all characters in the *Sidebar* is also filtered based on the scene and character type selection. It can be sorted according to outward engagement, outward sentiment, or inward sentiment, summed across only the selected scenes, in both descending and ascending order. Due to screen real estate constraints, the default list of characters shows only the top 5 characters from the sorted list, but the user can also expand the list to view all characters.

The nodes in the node-link graph are size-coded based on their overall engagement to all other characters in the selected scene/character type (i.e., filter window). The radius is linearly scaled by engagement, which is normalized across the selected window so that the minimum and maximum node sizes are the same for every network displayed. The nodes are also color-coded by character type. The categorical color scheme is derived from Vega. [4]

[4] https://vega.github.io/vega/docs/schemes/

The links are rendered as tapered curvatures to indicate the direction of the relationship. The smaller end is attached to the speaker and the wider end to the recipient. Although we explored several directed-edge representations, including standard arrows and multicolored links, we ultimately decided on a tapered representation based on the findings from Holten et al [4]. Since many pairs of nodes have a bidirectional relationship (i.e., one link from A to B and one from B to A), we used curved edges, rather than straight ones, to clearly distinguish and minimize overlap between incoming and outgoing links. Like the size of the nodes, the thickness of the links encode the character-to-character engagement of that particular relationship. Thickness is normalized across the selected filter window.

The links going out of a node are assigned colors based on the average sentiment valence of the relationship in the selected filter window. If only one scene is selected, then the sentiment is simply the sentiment for that scene. The colour of each link encodes the character's sentiment towards the other, using a five-bin red-blue diverging colour scheme: bright red (very negative), red, gray (neutral), blue, and bright blue (very positive). When no links are selected or hovered over, all links are light grey, as we chose not to display the sentiment encoding by default. This was deliberate as we felt like the node-link graph was too messy with all links coloured by sentiment.

The colours of the links are displayed via the two main interactive functionalities in this view. First, hovering: while the user hovers over a node, the links will be coloured according to the sentiment values. Second, selecting a node (by clicking): when the user clicks on a node, the links will be coloured and the outline of the selected node will be bolded, until the node is deselected (on another click).

4.2 Character-Pair View

We initially considered several possibilities for the *Character-Pair view*. Our first idea was to use scatter plots with shape-coded point marks (e.g., circle and triangle) for each character in a character pair, with the x-axis representing the scene number and a diverging y-axis representing the sentiment valence. The advantage of this approach is that the point mark can also be size-coded to encode the character-to-character engagement in each scene. This representation would enable the change in engagement across scenes to be conveniently presented. However, we realized that the sentiment valence data turned out to be very sparse (i.e., most character pairs only had one or two scenes with data), and therefore the choice of these design idioms would make it difficult to track changes in sentiment from scene to scene. This was especially difficult when dots were far apart (e.g., if characters interacted only in Act 1, Scene 1, and then in Act 5, Scene 1). We considered fixing this by interpolating between marks by drawing a line to better visualize the change in sentiment. However, because our sentiment data was discretized per scene, drawing a line across scenes with no data was misleading: it implied that there was some specific sentiment valence when in fact there was none.

In our final design, the *Character-Pair view* consists of small multiples of grouped bar charts. Each chart shows the sentiment valence at each scene, demonstrating the development of the relationship between a pair of characters throughout the play. It contains two grouped bars, encoding the two possible directions of a relationship, with the x-axis and y-axis encoding the same information as in the original design. The background of the charts are coloured in light grey for scenes in which the two characters have zero engagement. To support more dense small multiples, we decided to remove ticks from the x-axis which would have signified the scene number. Instead, we render small dotted lines in the background of each chart that delimit scenes.

A Midsummer Night's Dream consists of 71 two-way pairs of character interactions. The small multiples are sorted so that the most "interesting" (largest total absolute sentiment valence) ones are at the top. By default, character pairs that consist only of one-way interactions (i.e. A talks to B, but B never talks to A) are hidden, as they are less likely to provide interesting insights about the story. This reduces the list of small multiples to be rendered to 23.

After analyzing the sentiment valence values, we decided to visualize them without normalization. This caused a problem in determining

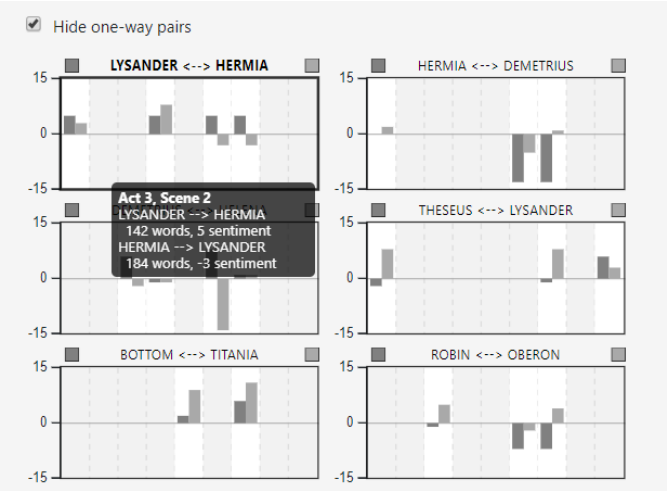

Fig. 2: The *Character-Pair view*. The six plots represent six pairs of characters with the highest total absolute sentiment valence. When the cursor is hovered over a chart, a tooltip pops up displaying relevant information like the scene of interest, sentiment, and engagement.

the range of the axis. Initially, we scaled the y-axis according to the minimum and maximum sentiment valence for each specific pair of characters. However, these dynamically scaled axes made it difficult to compare between different character pairs. Therefore, we decided to use only sentiment data calculated with the Bing Liu lexicon and fixed the y-axis for every pair to range from -15 to 15, which was calculated based on the overall minimum and maximum sentiment. We used a diverging axis with 0 at the middle as sentiment was also a diverging attribute, making it easy for the user to determine positiveness or negativeness of the value at a glance.

For each chart, the color of the left bars is dark grey, and the color of the right bars is medium grey. The titles of the bar charts are in the form $A \longleftrightarrow B$. The dark grey bars encode sentiment from A to B and the medium grey from B to A. Because the y-axis does not have fine-grained ticks, we added a tooltip, which allows the user to see the specific sentiment values that the bars represent, as well as the corresponding engagement, through a mouse hover. Figure 2 shows the first six plots in the *Character-Pair view* with the cursor hovering over the sixth column (Act 3, Scene 2) of the chart titled LYSANDER $\longleftrightarrow$ HERMIA.

4.3 Interactions Across Views

ShakesPeer has three interactions that propagate across views, which take the form of shared filtering and linked highlighting.

1. When a subset of scenes or character types are selected for filtering via the *Sidebar* menu, the *Character Overview*'s node-link graph and the *Character-Pair view* will be updated to include only characters that satisfy the filtering options.

2. When a character is selected in the *Character Overview*, only the charts of the selected character's relationships are shown in the *Character-Pair view*. If multiple nodes are selected, then the *Character-Pair view* shows the charts of all pairs with at least one selected character.

3. When a character is hovered over in the *Character Overview*, charts of the hovered character's relationships are highlighted in the *Character-Pair view*.

5 Implementation

ShakesPeer's implementation involves three key steps: (1) pre-processing the play script, (2) performing character-to-character sentiment analysis, and (3) building the web app itself. We first describe our methods for pre-processing and sentiment analysis (1+2), and then discuss the details of how we implemented the visualization and web app (3). The distribution of work across team members is presented in Table 3.

Table 3: Distribution of work across all team members, in percentages.

Task	Frances	Kevin	Mint
Data Pre-Processing	40%	0%	60%
Sentiment Analysis	0%	0%	100%
Character Overview Implementation	100%	0%	0%
Character-Pair view Implementation	0%	90%	10%
Sidebar Implementation	0%	70%	30%
Report and Presentation	30%	20%	50%

5.1 Pre-processing and Sentiment Analysis

The raw data from Folger Digital Texts is a text file containing a list of all the characters, act markings, scene markings, speakers of each speech, and the speeches themselves. First, we manually assigned a character type to each character, based on existing knowledge about the play's plot. Then, we extracted each speech with the assumption that the content of the speech is directed to the character spoken right before it [9]. However, we found that this method was inaccurate in many cases, especially in scenes where many characters are simultaneously present. Therefore, we went through the data and manually corrected the recipient of each speech to calculate the sentiment more accurately.

To perform sentiment analysis in Python 3, we experimented with two lexicons: AFINN [10] and Bing Liu [6]. AFINN has an existing Python library, which can score a speech (from a word to multiple sentences) by accumulating the score of each word. Bing Liu lexicon can be downloaded through the Natural Language Toolkit, but we needed to implement the scoring function by ourselves. While calculating sentiment analysis, we also calculated the engagement towards another character based on the number of words in each speech directed to that character. Finally, we stored this derived data in a TypeScript dictionary. The key of the dictionary is a string in the form "A-B", where A and B are names of two different characters. The value of the dictionary is a list of tuples, where each tuple corresponds to a scene, and the values in the tuples are the sentiment from A to B and the engagement of A to B in that scene.

5.2 Visualization and Web App

The visualization loosely follows the popular MEAN stack [5] for building web apps. The back-end, built with Node.js [6] and Express [7], is responsible for statically serving the front-end components to be viewed in the browser. Our original intention was to follow the MEAN stack and use MongoDB [8] as our database solution. However, due to time limitations and to support prototyping, we decided to instead load our pre-processed data directly into memory as Javascript objects. A complete pipeline would involve storing the pre-processed data in MongoDB, and having the back-end serve the data through endpoints to the front-end.

The front-end was built with the AngularJS framework [9] and styled with Bootstrap [10]. The visualization components were built directly with D3.js [11]. We did not use any additional libraries or toolkits that build on or extend D3.js. Using the Angular framework, we split our application into 3 major components: *Sidebar*, *Character Overview*, and *Character-Pair view*, corresponding to the main views of our visualization. The implementation details for the *Character Overview* and *Character-Pair view* components will be described in further detail below.

[5] `http://meanjs.org/`
[6] `https://nodejs.org/en/`
[7] `https://expressjs.com/`
[8] `https://www.mongodb.com/`
[9] `https://angularjs.org/`
[10] `https://getbootstrap.com/`
[11] `https://d3js.org/`

In addition, we also implemented an Angular service containing state information that's shared and can be updated across all components with RxJS's [12] implementation of the observer design pattern [1]. State information includes filtering options from the *Sidebar* and selected/hovered nodes (characters) from *Character Overview*. This was necessary for implementing features like linked highlighting and shared filtering across components.

5.2.1 Character Overview

The nodes and links are rendered as SVG elements with D3.js. The tapered, curved geometry of the links are generated by rendering two quadratic Bezier curves with shared endpoints. The width varies along the length of the link to indicate direction, and the distance between the two curve peaks is proportional to the engagement value of the relationship it encodes. To keep the nodes centered in the visible area while preventing node overlap, we used D3's force-directed graph layout implementation. Two forces were specified: (1) a repulsive force between the nodes, and (2) a small positive charge force towards the center of the view.

5.2.2 Character-Pair View

The small multiples in the *Character-Pair* component were generated by using D3's `selection.each` function to create a scope for each multiple, effectively making one SVG element per graph, and defining local data values as appropriate. Because we needed to render a bar chart with a vertical diverging scale, we implemented it with limited examples as most had diverging bar charts that were horizontal. We also experienced challenges in working with `selection.each` as our inexperience with scope changes made it difficult to implement functionalities that would be trivial otherwise.

Most work was spent in making sure all graphs rendered correctly in relation to each other, as well as in filtering and sorting the small multiples. Sorting of the small multiples was based on the total magnitude of sentiment valence values for each chart. Charts were arranged based on total sentiment magnitude in descending order, from left-to-right, then top-down.

6 Results

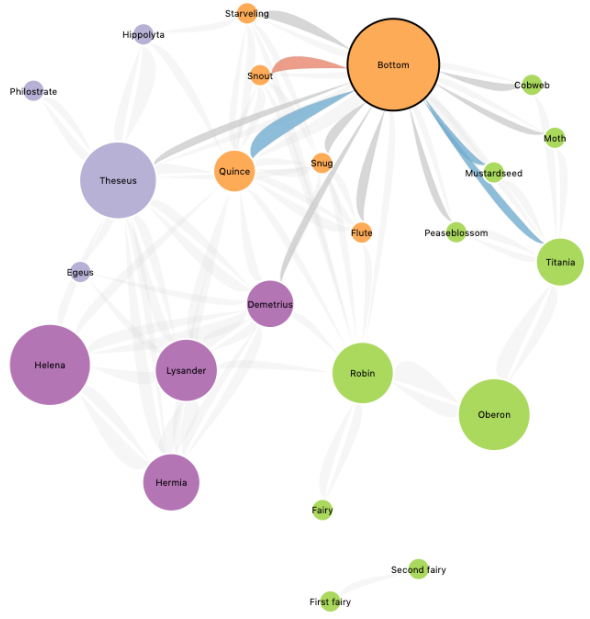

Fig. 3: The *Character Overview* with Bottom selected.

[12] `https://rxjs-dev.firebaseapp.com/`

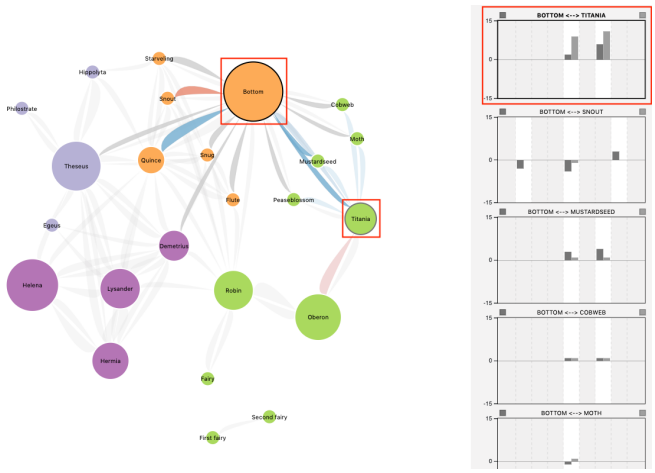

Fig. 4: Linked highlighting between the *Character Overview* and the *Character-Pair View* when hover over the node labelled 'Titania'.

In this section, we discuss a scenario of use from the perspective of Sarah, a graduate student who is writing a character analysis for an English literature class. The analysis is centered around the character of Bottom, the self-possessed actor who is a central figure in *A Midsummer Night's Dream*.

After an initial read of *A Midsummer Night's Dream*, Sarah opens the ShakesPeer web application. In the *Character Overview*, she sees a dense network of all interacting characters in *A Midsummer Night's Dream*. Since she is specifically interested in the character of Bottom, Sarah selects the node labelled 'Bottom'. Upon selection, the node is highlighted with a thick black stroke. The outgoing edges, each pointing to a character that Bottom speaks to at some point in the play, are colour-coded based on sentiment. From a quick glance, she can see that Bottom's dialogue with these characters is mostly neutral or positive in sentiment (Fig. 3).

In her analysis, Sarah would like to write a paragraph about Bottom's relationship with Titania. With Bottom selected, she hovers over the node labelled 'Titania'. This highlights the bar chart in the Character Pair View that visualizes the sentiment between the two characters throughout the play (Fig. 4). Despite the importance of their relationship in *A Midsummer Night's Dream*, she discovers that Bottom and Titania appear together in only two scenes: Act 3, Scene 1 and Act 4, Scene 1. The height of the bars indicate that Titania has a stronger positive sentiment towards Bottom compared to his sentiment towards her (Fig. 5). Sarah finds this intriguing. With this knowledge, she returns to the text and discovers that despite Bottom's self-possession, he conveys a subtle skepticism during his encounters with Titania. For instance, when Titania declares her love for him in Act 3, Scene 1, Bottom replies that she "should have little reason for that" [12].

(a) Act 3, Scene 1 (b) Act 4, Scene 1

Fig. 5: A grouped bar chart showing bidirectional sentiments by scene between Bottom and Titania. In both figures, a tooltip appears when hovering over the area outlined in red.

7 Discussion and Future Work

In this section, we will first discuss one major challenge that we faced during the design process. Then, we will discuss the strengths, weaknesses, and limitations of *ShakesPeer* in regards of the four tasks we planned to support described in Section 3.3. Then, we will describe the features we were unable to implement due to time constraints as well as possible extensions from the current design.

7.1 Challenges

In the initial stages of the design process, we chose our visual encoding idioms prior to knowing what our sentiment data would look like. After deriving the sentiment values, we were surprised to see that the data was very sparse overall. Additionally, for the pairs with sentiment values, the inter-pair variance was unexpectedly high. We had to go undergo extensive revisions for the final design and task abstraction to account for the sparsity and variance of the data. For instance, we realized that it was important to differentiate the pairs without any sentiment data from the pairs with sentiment data. In retrospect, we believe that it would have been extremely helpful to go through some iterations of rapid prototyping before committing to a representation. During the revision process, we found that it was easy to quickly plot graphs of our sentiment data using matplotlib [13] library in Python 3 via Jupyter notebook.

7.2 Strengths, Weaknesses, and Limitations

Considering the complexity and scale of Shakespeare's plays, *ShakesPeer* helps users find the "interesting" parts or aspects of the play quickly (**T3**). The *Sidebar* provides a comprehensive filtering functionality for the *Character Overview*, allowing the user to select a subset of the play that they would like to see, both in terms of scenes and characters, to avoid the "hairball effect". The plots in the *Character-Pair view* are also sorted by the magnitude of sentiment in the plot, making it easy for users to discover the plot of their interests. This functionality is important for expanding our work to include other Shakespeare's plays, since most of them have even more characters than *A Midsummer Night's Dream*.

Our data abstraction from plain text to two numerical attributes, engagement and sentiment, greatly simplifies the data while still allowing the user to gain important information about character relationships, for the discover (**T1**), compare (**T2**), and summarize (**T4**) tasks. However, the exploration of raw content of the play, such as explaining the cause of sentiments or the content of the engagement, is beyond the scope of our task abstractions. If the user wants to relate the displayed information to the actual story plot, they will have to do a look-up by scene, based on the tooltips in *Character-Pair view*.

Another problem with our data abstraction is the naive approach we employed for determining recipients of sentiments. Our methods will work better for some plays compared to others. For example, it will work better for a play with a lot of direct confrontations compared to a play with a lot of gossip (i.e., dialogue between characters about a different character who is not a part of the conversation). Therefore, the sentiment valence shown in the plot should be interpreted carefully in all tasks (**T1-T4**).

As mentioned briefly in Section 4, there are several trade-offs in the design decisions we made, which result in three limitations of the tool. First, in the *Character Overview*, the thickness of the links encoded engagement, meaning that they take up a considerable amount of space. Inevitably, this causes the links to overlap. This is especially problematic when a thinner link is completely occluded by a thicker link. We tried to alleviate this problem by reducing the opacity of the links, so that even when they are overlapped, you can still make out the thickness of the parts that are covered. In the case above, where a thinner link is completely covered, this approach allows the user to view the thickness of the thinner link, although it would still be difficult to determine the color of the link. A workaround is if the user manually drags around the nodes to rearrange the links by themselves. Nevertheless, this could be an inconvenience to the user when performing compare (**T2**) and summarize (**T4**) tasks.

Second, in the *Character Pair view*, we chose not to normalize the sentiment valence values across displayed plots and did not allow axis rescaling. Therefore, small sentiment values, which might still be important, cannot be easily distinguished or compared. We decided on this design choice because we thought that having all plots under the same scale makes for easier comparison at a glance, and we tried to alleviate this problem by providing the tooltips, which allows the users to see more specific information. Depending on the specific information or pattern that the user wants to observe, this could be a limitation for the compare task (**T2**).

Finally, with the bar chart design in the *Character-Pair view*, we decided to omit the visualization of engagement in each scene in order to show the sentiment valence values more clearly. We made this design decision because we prioritized sentiment information over engagement. We tried to accommodate users who wished to look at engagement by encoding the information in the *Character Overview* as well as by using the Bing Liu lexicon, where the sentiment values have more correspondence to the number of words in the speech than the AFINN lexicon. However, this is still a major limitation for discovering how engagement between a pair of characters might change over time (**T1**).

7.3 Future Work

Due to time constraints, only one play, *A Midsummer Night's Dream*, was visualized using our tool. In the future, we would like to evaluate the generalizability of *ShakesPeer* for other plays, including those with denser, more complex character-relationship networks. There are also two additional features that we would like to include in future iterations of *ShakesPeer*: (1) the capability to compare character relationships across different plays, and (2) the option to view sentiment values from different sentiment lexicons. We found that in some cases the sentiment values from the Bing Liu lexicon were significantly different from the results from AFINN. Thus, it could be useful to give users the option to view the sentiment values from their preferred lexicon.

Usability testing is a critical component of the design process to determine whether a new product or tool is useful and easy to use for its intended purpose [11]. Ultimately, we would like to evaluate *ShakesPeer* with representative users, such as students, literary scholars, and professors. Gathering user feedback will help us identify any issues with the design and will provide insight into the ways that the tool can be improved. For instance, there may be other measures of interest besides character-to-character sentiment and engagement that users want visualized, such as themes and settings of speech.

8 Conclusion

We propose *ShakesPeer*, an interactive tool for visualizing detailed character and relationship information (i.e., sentiment and engagement) based on the dialogue of a play. *ShakesPeer* is comprised of two main views. The *Character Overview* provides a summarized visualization of all character relationships in selected scenes, and the *Character-Pair view* shows the development of sentiment and engagement between pairs of characters throughout the story. With the *Sidebar*, users can isolate characters of interest by filtering the visualizations by scene, character, and character type. Although *ShakesPeer* currently only visualizes *A Midsummer Night's Dream*, it is generalizable and extensible to other Shakespearean plays. The immediate next step for *ShakesPeer* is to derive requirements for future iterations by conducting usability tests with representative user groups.

Acknowledgments

The authors wish to thank Dr. Tamara Munzner for her guidance throughout this project. We also would like to thank Marjane Namavar, Matheus Stolet, and Vaastav Anand for their valuable feedback and suggestions during the peer project review sessions.

[13] https://matplotlib.org/

REFERENCES

[1] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns: Micro-Architectures for Reusable Object-Oriented Design*. 1994.

[2] B. Gaydin. Digital tools for comparative thesaurus analysis of russian translations of w. shakespeare's works: Results of the first year. *Horizons of Humanities Knowledge*, (6):169–182, 2017.

[3] M. Grandjean. Network visualization: Mapping shakespeare's tragedies. `www.martingrandjean.ch/network-visualization-shakespeare/`, 2015.

[4] D. Holten, P. Isenberg, J. J. Van Wijk, and J.-D. Fekete. An extended evaluation of the readability of tapered, animated, and textured directed-edge representations in node-link graphs. In *2011 IEEE Pacific Visualization Symposium*, pp. 195–202, 2011.

[5] E. Johansson. Tolkien's books analysed.

[6] B. Liu. Opinion mining and sentiment analysis. In *Web Data Mining*, pp. 459–526. Springer, 2011.

[7] S. Mohammad. From once upon a time to happily ever after: Tracking emotions in novels and fairy tales. In *Proceedings of the 5th ACL-HLT Workshop on Language Technology for Cultural Heritage, Social Sciences, and Humanities*, pp. 105–114. Association for Computational Linguistics, 2011.

[8] T. Munzner. *Visualization Analysis and Design*. AK Peters Visualization Series. CRC Press, 2015.

[9] E. T. Nalisnick and H. S. Baird. Extracting sentiment networks from shakespeare's plays. In *2013 12th International Conference on Document Analysis and Recognition*, pp. 758–762. IEEE, 2013.

[10] F. Å. Nielsen. A new ANEW: Evaluation of a word list for sentiment analysis in microblogs. *In Proc. ESWC-11*, 2011.

[11] J. Nielsen. *Usability engineering*. Elsevier, 1994.

[12] W. Shakespeare. *A Midsummer Night's Dream*. Ginn and Company, 1910.

[13] R. Zakovich. Shakespeare: A data visualization. https://www.informationisbeautifulawards.com/showcase/2044-shakespeare-a-data-visualization.