ShakesPeer: A Tool for Visualizing Character Relationships in Shakespearean Literature

Frances Sin, Kevin Chow, Mint Tanprasert
francsin@cs.ubc.ca, kchowk@cs.ubc.ca, tt1996@cs.ubc.ca

Introduction

The works of William Shakespeare have influenced generations of literature and theatre and continues to inspire modern creatives. In particular, Shakespeare is praised for his unparalleled ability of characterization. Shakespearean characters are richly textured and have distinctive personalities and motivations. The interactions between these characters play an important role in developing various plots and themes.

We propose a new tool, ShakesPeer, which will visualize relationships between Shakespearean characters. The basic idea of this approach is to reveal useful and nuanced insights about characters' relationship development over time. This can result in a better understanding of the plot, in particular related to the characters, and provide a starting point for deeper literary analysis.

Our team consists of three Master's students with strong HCI backgrounds. In addition to HCI expertise, we have experience in areas such as data manipulation and visualization, academic research, and programming. The core motivation for this project originates from Mint Tanprasert's personal interest in Shakespearean literature.

Previous Work

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* project, which provides complete TEI-annotated Shakespearean plays, and Open Source Shakespeare (OSS) [1], which provides statistical features and keywords search on Shakespearean texts.

Shakespearean 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 [4], and to explore a certain part of the text, taking into account its positioning in the whole text [5]. Word-based visualization is mostly done in the form of speech distribution visualization. Zakovich orders the speech distribution per scene of 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 scatterplot of words, showing their correlations and similarity [10]. Character-based visualization is mostly node-link networks of character co-occurence in a scene. Grandjean creates such network for every Shakespeare's tragedies, allowing high-level comparison of network structures between stories [5].

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 create a node-link network of characters where each link is color-coded as green, if the two characters share "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 [6]. 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 [8].

Domain, Data and Task

We will initially focus on *A Midsummer Night's Dream* as an exemplary piece of Shakeseparean 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 *A Midsummer Night's Dream* 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 a tool like ShakesPeer. However, if time permits, we would like to also include other pieces of Shakespearean literature and support comparisons of character relationships across texts.

Our dataset will consist of the full script of *A Midsummer Night's Dream*, collected from Open Source Shakespeare [1]. *A Midsummer Night's Dream* has 5 acts with 9 scenes. There are 2 scenes per act, except for the last act (Act 5), which only has one scene. There are 605 speeches, 2290 lines, and 16,511 words in total, 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 monologue.

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 overall sentiment and character speech frequency.

Character Relationship Network

In the character relationship network, the nodes are characters, and the links between them encapsulate sentiment, or characters' emotional valence towards one another, and co-occurrence, or the presence of both characters' speech within a scene. Each node has a character name attribute, which is categorical with 22 unique levels, each corresponding to one of the characters in the story.

Each link has three types of attributes: sentiment of the current character towards the other, sentiment of the other towards the current character, and the set of scenes in which there is co-occurrence. Both directions of sentiment (current→other and other→current) are captured as ordered, quantitative, but diverging attributes, with a potential range of negative infinity to positive infinity, where negative numbers are negatively valenced and positive number are positively valenced. Character-to-character sentiment will be calculated automatically based on a method proposed by Nalisnick and Baird [9], on a per-scene basis.

We will capture each scene in the set as an ordinal attribute, which ranges from 1 (Act 1, Scene 1) to 9 (Act 5, Scene 1). For example, the set of co-occurrence scenes between Oberon and Titania, two characters in *A Midsummer Night's Dream*, may look like: [1, 2, 4, 8], meaning that they appear in Act 1, Scenes 1 and 2, Act 2, Scene 2, and Act 4, Scene 2. There can also be an attribute derived from the set of scenes, which is simply the total number of co-occurrence scenes between the two characters. This derived attribute is ordered, quantitative, and sequential, ranging from 0 to 9 total scenes. For the example above, this would be 4 co-occurrence scenes.

Scene Table

Scene (key)	**Act**	**Sentiment**	**Oberon**	**Titania**	*(other characters' speech frequency attributes)…*
1	1	53.95	251 words	530 words	
2	1	70.41	104 words	20 words	
(other scene items....)					

Table 1: An example subset of our data abstraction in the scene table. Columns with character names (column 4 and onward) represent that character's scene speech frequency, in number of words.

Information for each scene is captured in a simple flat table (see Table 1), where each row of the table is represented by a scene in the story. The key of the table is an explicit scene attribute, similar to the one in the character relationship network. The value attributes of the table include the act of the scene, overall scene sentiment, and speech frequency per character in the scene. Because there are 22 characters in *A Midsummer Night's Dream*, each row has 24 value attributes in total. The act attribute is an ordinal attribute that ranges from 1 (Act 1) to 5 (Act 5). Again, like in the network, the overall sentiment is an ordered, quantitative, but diverging attribute, ranging from negative to positive infinity. Unlike character-to-character sentiment, overall sentiment is calculated from all of the words in the scene, instead of just speech between character pairs. For each character, we calculate their speech frequency within the scene by simply summing up the number of words they have spoken. Thus, these attributes are all ordered, quantitative, and sequential, which may range from 0 to 16,511 (the total number of words in *A Midsummer Night's Dream*).

Task Abstraction

ShakesPeer will support the following tasks:

- Discover how sentiment changes over time between two characters.
- Discover how a character's speech frequency changes over time.
- Compare the relationship between one or more pairs of characters in terms of sentiment and speech frequency.
- Identify points in the story where key events lead to major changes in relationships.
- Present an overview of all the characters' relationship towards one another.
- Enjoying the complexity of Shakespearean literature through compelling and interactive visualizations.

Proposed InfoVis Solution

The user will use the visualization tool in two main views, the main view (Figure 1) and the relationship view (Figure 2). In the main view, there will be a menu panel on the left side of the screen, allowing the user to select a subset of characters and the scene that they want to analyze. The list of scene will be encoded with colors, indicating the scene's overall sentiment, allowing the user to quickly identify significant changes and development in the story.

Each character will show up as a node, and a link between two nodes indicate co-occurrence of the two characters in the same scene. If the scene is not selected in the menu, the visualization will be for the whole story. That is, the thickness of the link will encode the number of scenes with co-occurrence. The area of a circular node will encode the number of words spoken by the character throughout the story. If the scene is selected, all links will have the same thickness (the existence of the link indicates co-occurrence in the scene) and the area of the node will indicate the number of words spoken by the corresponding character in that specific scene. In both scenarios, when the user hovers over node, all links going out from that node will be highlighted with colors that correspond to the overall sentiment (in a scene or in a whole story) of the pair of characters.

When the user selects an edge in the main view, the relationship view appears as a new window on the right side of the screen. If many edges are selected, these windows will be vertically stacked into a column. Each window shows the speech frequency of the two characters across the story as two line charts overlaid on top of each other. Each line is color-coded to indicate the character. The background of the line charts will also be color-coded to display sentiment analysis for every scene across the story, allowing the user to see how the relationship between the two characters changes across time. Selecting one window in the relationship view also has the effect of highlighting the corresponding edge and the two nodes on that edge in the main view.

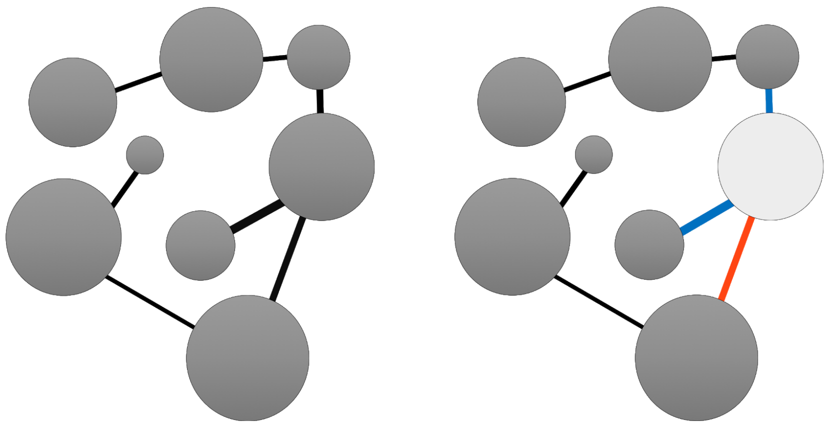

Figure 1: (left) The main view of ShakePeer with no nodes selected. (right) The main view with one node selected. The outgoing edges are coloured based on the selected character's sentiment towards each character he/she interacts with in the play.

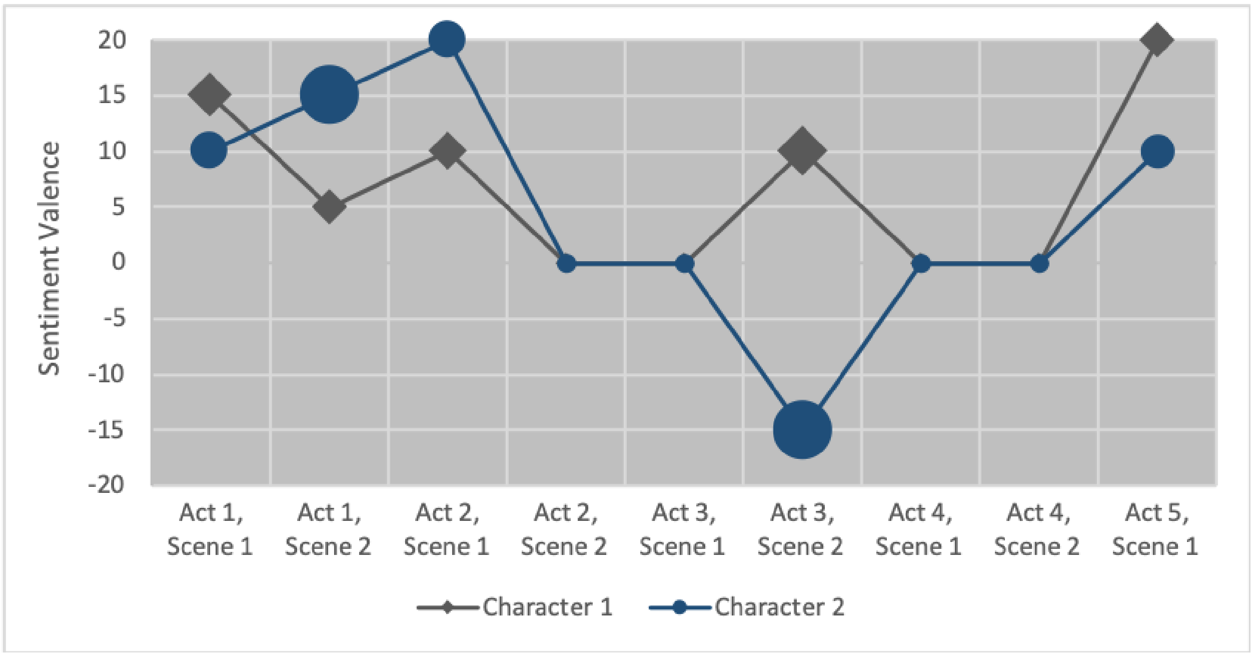

Figure 2: The relationship view of ShakesPeer, which displays the characters' sentiment throughout the play. The size of the markers encodes the total number of words spoken by a character in each scene.

Scenario of Use

Stuart is a first-year university student working on an essay analyzing the theme of magic in Shakespeare's *A Midsummer Night's Dream*. After an initial reading of the play, he opens the ShakesPeer web tool and loads the main view. He sees a node-link network which maps every relationship between characters in *A Midsummer Night's Dream*. For his essay, Stuart wants to analyze the character of Puck, who is a fairy that uses magic to cause turmoil and chaos in the story. He hovers over the node for Puck and is surprised to see that the edge between Puck and Oberon is red. Stuart does not recall any negative interaction between these characters. He clicks on the edge, which opens the relationship view on the side of the screen. He sees that Puck and Oberon's relationship at the start of the story is positive (both characters have positive sentiment towards each other), but in Act 3, Oberon's sentiment towards Puck is very negative. When Stuart re-reads Act 3, he develops a better understanding of what is being stated by each of the characters. He realizes that Oberon is angry towards Puck because he learns that Puck had mistakenly smeared the love potion on Lysander's eyes instead of Demetrius's. Stuart had initially misunderstood the overall tone of the act due to Shakespeare's unconventional sentence structures and phrasing. Stuart returns to the main view and looks at the edges going out of Puck. He sees that Puck interacts with every character involved in the main romantic entanglement in the story. When he clicks on these edges, he notices that most of the interactions occur in Act 3 Scene 2, which is when a mix-up of lovers causes animosity among the main characters. This affirms his conclusion that magic plays a vital role in *A Midsummer Night's Dream*, particularly as it relates to love and relationships.

Proposed Implementation Approach

We will build ShakesPeer using D3.js [3], with the intention that it serves as a standalone webapp. Text-based analysis and cleanup, such as for sentiment, will be done with Python, as there are a plethora of libraries and resources that are available to do so. An option is to use the MEAN [7] stack for the webapp, which involves using MongoDB as the database, Express, AngularJS, and Node.js. We have prior experience with this popular JavaScript software stack. Once the full play script is cleaned up and attributes are derived in Python, we will populate a MongoDB database to allow us easy access to the data.

Milestones and Schedule

Task	**Est. Hours**	**Deadline**	**Description**	**Who?**
Data Cleaning/Basic Transformations	10 hours	Nov. 15	Download and clean-up dataset (play script). Calculate speech frequency and extract character texts.	FS/KC
Set up Basic Demo	10 hours	Nov. 15	Set up group logistics; GitHub, figure out any boilerplate code for getting web demo up and running.	KC
Sentiment Analysis	25 hours	Nov. 17	Research about character-to-character sentiment analysis is done; implement algorithm and run on dataset.	MT
Main View	20 hours	Nov. 18	Implement main view (node link visualization, sidebars, etc.).	FS
Peer Project Review 1	6 hours	Nov. 19	Prepare slides and initial small demo.	Everyone
Miscellaneous Implementation	15 hours	Nov. 30	Filtering, scene- or character- based, additional sidebar features.	FS
Detailed View	35 hours	Dec. 1	Implement detailed view. Link to main view. Enable multiple comparisons of pairs by stacking views.	KC/MT
Paper Outline	10 hours	Dec. 1	Complete paper outline in bullet-point form.	Everyone
Peer Project Review 2	6 hours	Dec. 4	Prepare slides and more detailed demo.	Everyone
Demo Polishing/Post Review Changes	10 hours	Dec. 8	Polish demo, animations? Any changes needed from peer project review.	Everyone
Final Presentation	20 hours	Dec. 10	Prepare slides, demo, potential video. Rehearse for presentation.	Everyone
Final Paper	25 hours	Dec. 13	Finalize and edit paper. Create all necessary tables and figures.	Everyone

References

[1] “A Midsummer Night's Dream | Open Source Shakespeare.” George Mason University, 2019, http://www.opensourceshakespeare.org/views/plays/playmenu.php?WorkID=midsummer.

[2] Correll, Michael, and Michael Gleicher. "What Shakespeare taught us about text visualization." *IEEE Visualization Workshop Proceedings: The 2nd Workshop on Interactive Visual Text Analytics: Task-Driven Analysis of Social Media Content*. 2012.

[3] Bostock, Mike. “D3.js - Data-Driven Documents.” 2019, https://d3js.org/.

[4] “Digital Tools for Comparative Thesaurus Analysis of Russian Translations of W. Shakespeare's Works.” Version Variation Visualization - Project, www.delightedbeauty.org/vvvclosed/home/project.

[5] Grandjean, Martin. “Network Visualization: Mapping Shakespeare's Tragedies.” Martin Grandjean, 23 Dec. 2015, www.martingrandjean.ch/network-visualization-shakespeare/.

[6] Johansson, Emil. “Tolkien's Books Analysed.” The Lord of the Rings Family Tree Project, 2012, http://lotrproject.com/statistics/books/sentimentanalysis.

[7] “MEAN.JS - Full-Stack JavaScript Using MongoDB, Express, AngularJS, and Node.js.” 2014, http://meanjs.org/.

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

[9] Nalisnick, Eric T., and Henry S. Baird. "Extracting sentiment networks from Shakespeare's plays." *2013 12th International Conference on Document Analysis and Recognition*. IEEE, 2013.

[10] Zakovich, Rody. “Shakespeare: A Data Visualization.” Shakespeare: A Data Visualization - Information Is Beautiful Awards, 2017, www.informationisbeautifulawards.com/showcase/2044-shakespeare-a-data-visualization.