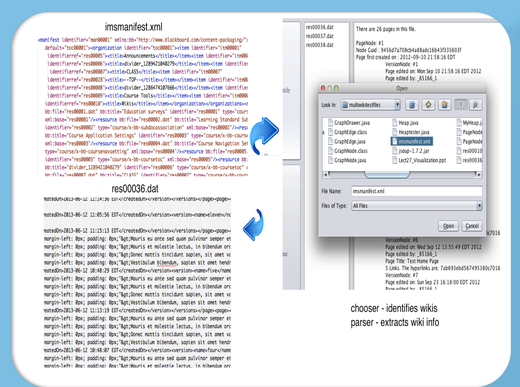
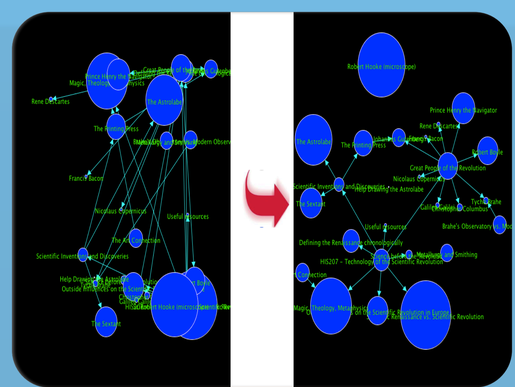
Wiki Visualizer for Blackboard

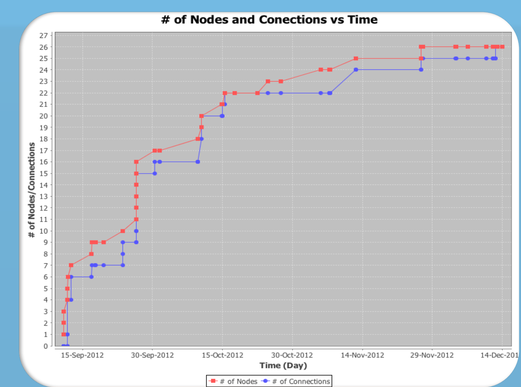
Ru Zhao (Advisor: Alistair Kwan.PhD)
University of Rochester

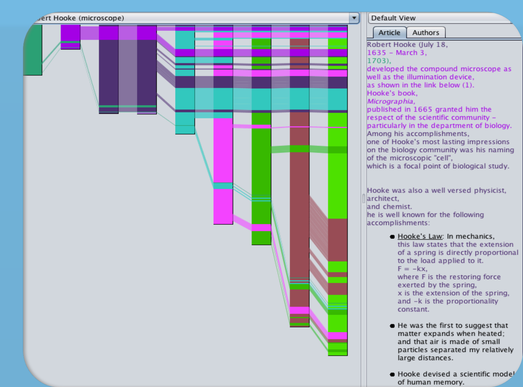
Introduction

Blackboard has a feature that allows students and professors to cooperate in creating wikis. These wikis contain a lot of information about the behaviors of students and the social dynamics of an academic community. These valuable data could be used as academic studies for the improvement of teaching and learning methods as well as allowing administrators to facilitate a better academic environment. The problem is the difficulty of extracting useful information out of these wikis for scholars to study. This app is created for the purpose of conveying the data using a visual medium and defining useful metrics.









Chooser & Parser

Problem: The wiki files and other class related data can be downloaded to a desktop. However, it is difficult to find the wiki files contained in the folder.

Solution: The xmlmanifest.xml is generated after the files are downloaded and it contains all the information regarding the downloaded items. The chooser parses the xmlmanifest file and returns the list of wikis contained in the folder.

Problem: The wiki files are also encoded in xml with the article contents being encoded in processed html.

Solution: The parser parses the wiki, extracts the article name, html content, time of creation and every individual edits. It then constructs a data structure that represents the articles and the links between them with time as a dependency for later manipulation.

Wiki Visualizer

Problem: The data is all meaningless without providing insight to the viewer. A popular way to visualize social networks is to represent people as nodes and relations between them as links in a graph. Wikis can be represented in a similar fashion, articles can act as nodes and hyperlinks between them can act as edges. Location of the nodes needs to be determined to present a visibly intuitive and graphically meaningful observation to the viewer.

Solution: The visualizer uses a force directed algorithm (force atlas) that treats the nodes as electrically charged particles that repel each other based on Coulomb's Law. The edges between the nodes behave like springs under Hooke's law and pulls the nodes closer. The results is a minimal energy state with the less connected nodes being further apart and highly connected nodes grouped together. The visualizer also adds a central attractor with a gravity scale that keeps the nodes from drifting off the screen. Damping scaling also added to provide the option of letting the user to choose between the tradeoffs of speed vs accuracy. Some nodes may get stuck in local minima, nodes can be dragged manually to reposition by user. The viewer can gain valuable information from this graph such as what topics are students most interested in, what topics are more related to the other, and how the wiki grows, etc.

Metrics & Plotter

Problem: Intuitive graphs are great, but it is still necessary to define useful metrics when conducting research.

Solution: Some useful metrics are node count, connection count, graph diameter, node centrality, there are many other important unexplored metrics that can provide useful insights to the user. The user gets a list of these metrics and receives the info upon request. More plans to further develop this.

Problem: There are many nice metrics graph creator out there such as Excel, Open office spreadsheet, Data Studio. The user can just receive the info and plot it on some other program. But the app is designed to be stand alone for user convenience.

Solution: To have a reason for plotting graphs on the app, the graphs should be attractive and appealing. The Plotter incorporates JFreeChart, an open source library that generates very aesthetically pleasing graphs.

Article Analyzer

Problem: Blackboard keeps a record of all the individual edits of an article page with time stamps. These info on individual articles can be extremely useful. For example, if a participant post something very controversial on the article page, the community will respond. It is then possible to classify what information are agreed upon and what subjects spark the most controversy. Other uses include identifying malicious individuals and plagiarizers.

Solution: Inspired by IBM research's *history flow*, the article analyzer assigns unique colors to each author and connects persistent text represented as chunks of the bars.

Problem: Blackboard only shows article edits by time stamps but does not show changes between two edits or what author contributed to which part of the text. Identifying them is a difficult task.

Solution: The article analyzer incorporates *Myer's Diff* algorithm to compare texts by finding the longest common subsequences, matching them as by the same author and as persistent data.

Problem: Currently the diff algorithm takes phrases separated by punctuations as tokens. A simple spelling correction will be taken as a new text. I plan to correct this with the *Levenstein Distance* formula.