Design Documentation

CS325: MedTop - An Interface for Visualization and Exploration of Medical Reflective Writings for Clinical Educators

Sean Kotrola

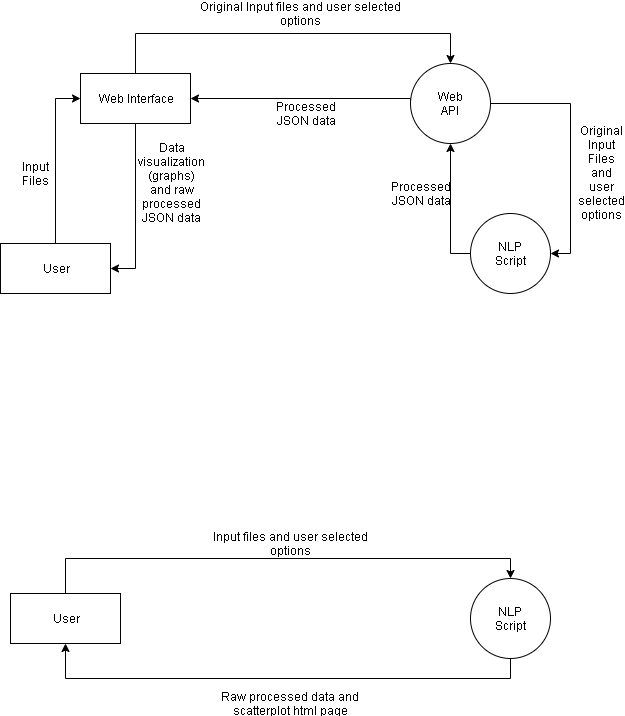
Aidan Myers

Suzanne Prince

Overview:

Current State:

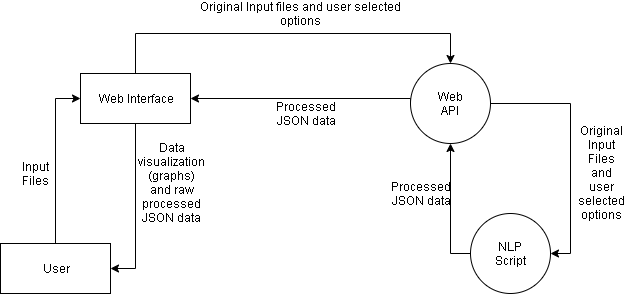
Reflective writing aids medical students in reflecting on their experiences while in medical school. These writings provide a wealth of information that could be used by medical educators to identify key challenges, improve the coverage of their curriculum, and promote interprofessional skills. However, extracting this information presents a challenge as manually reviewing these texts is labor intensive and difficult for humans to categorize given the variety of topics that can be discussed in a single survey question.To aid in this, the VCU’s Wright Center for Clinical and Translational Research developed a Natural Language Processing (NLP) tool that automatically extracts common challenges experienced by medical students from these reflective texts. This brings us to the current state, which can be represented in a data flow diagram.



*Figure 1: Current State Data-Flow diagram*

Future State:

However, utilization of this tool requires a level of programming knowledge not commonly held by medical educators. The goal of this project is to develop MedTop (Medical Topic analysis), a web application that will allow non-technical users to run and *fully explore* the results of this NLP tool without the need for any prior programming knowledge. This web application can be broken into two distinct parts. The web interface (front end) and the web API (back end).



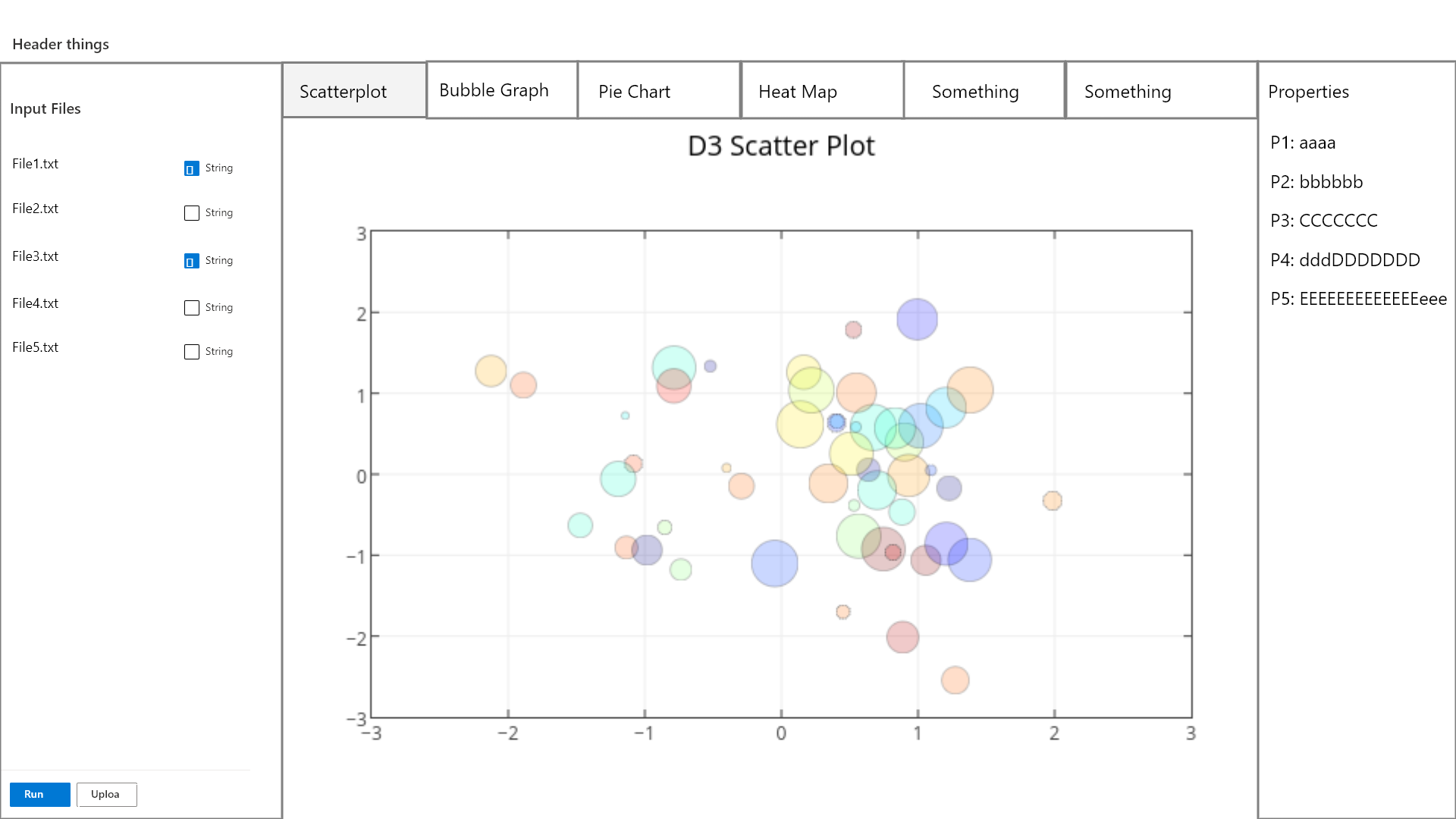
*Figure 2: Future State Data-Flow diagram*

Design

Front-end Overview:

The front-end will be built using React and D3.js. React will be used in implementing the user interface, while D3.js will be used to provide varied forms of interactive data visualizations. The main panel will display dynamic visualizations of results generated by the tool, and will be scalable. The left-side panel will provide management tools for uploading the data, including an interactive list of files, and will allow the user to easily search through and manage their files. This panel will also include a tab with settings for executing the analysis. The right-side panel will be populated with detailed information on data points the user has selected from the visualization, allowing them to easily interpret the information being displayed. The user will have the option to export their raw data, as well as any graphs generated. They will be able to re-import data, which will regenerate the visualization without having to reprocess data.

Web app (front-end) sketch:



*Figure 3: Web-app Visualization*

Back End Overview:

The back-end will make use of a Flask API, and will facilitate interaction with the NLP tool. Our backend does not utilize a database. Instead, we dynamically generate data based on user input and return it to the web app. This means that no data is ever stored, only processed and returned.

Our API consists of a single endpoint defined as “/runScript”. This endpoint is passed essential data needed to run the script, and outputs the processed data produced by the NLP tool.

API Design:

|  |  |  |  |
| --- | --- | --- | --- |
| Endpoint Name | Methods | Input | Output |
| /runScript | POST | inputFiles: Array of file objects to be processed  scriptOptions: JSON containing user selected options | result: JSON containing the processed information |

List of NLP tool options:

inputFile

Tfidfcorpus

wordVectorType

w2vBinFile

Outputdir

Prefix

windowSize

goldStandard

Threshold

dimensions

Scatter\_plot

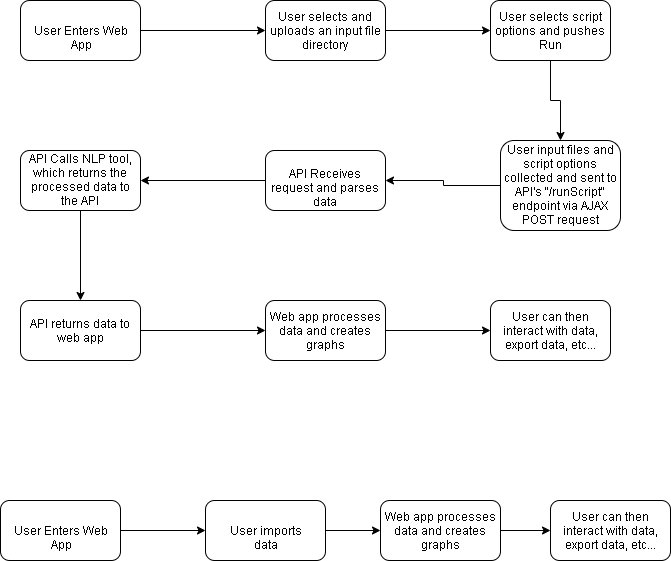
Umap\_neighbors

distmetric

Process Flow:

Scenario 1:

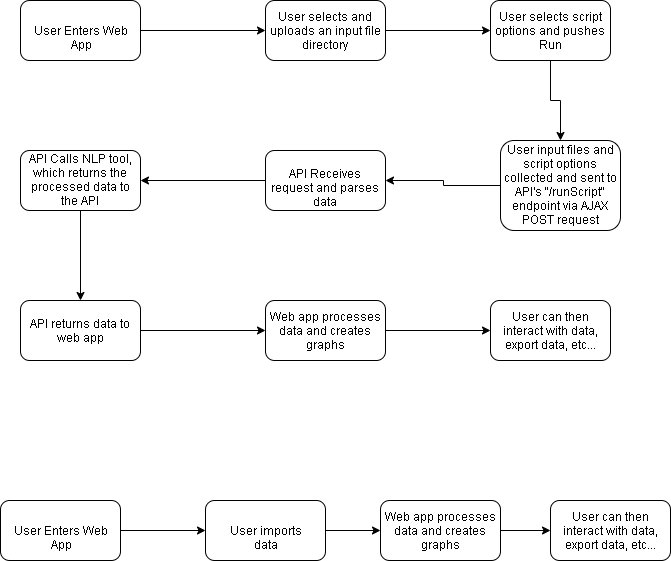
User uploads input files and selects options to run the NLP tool.



*Figure 4: Process flow scenario 1*

Scenario 2:

User imports previously exported data.



*Figure 5: Process flow scenario 2*