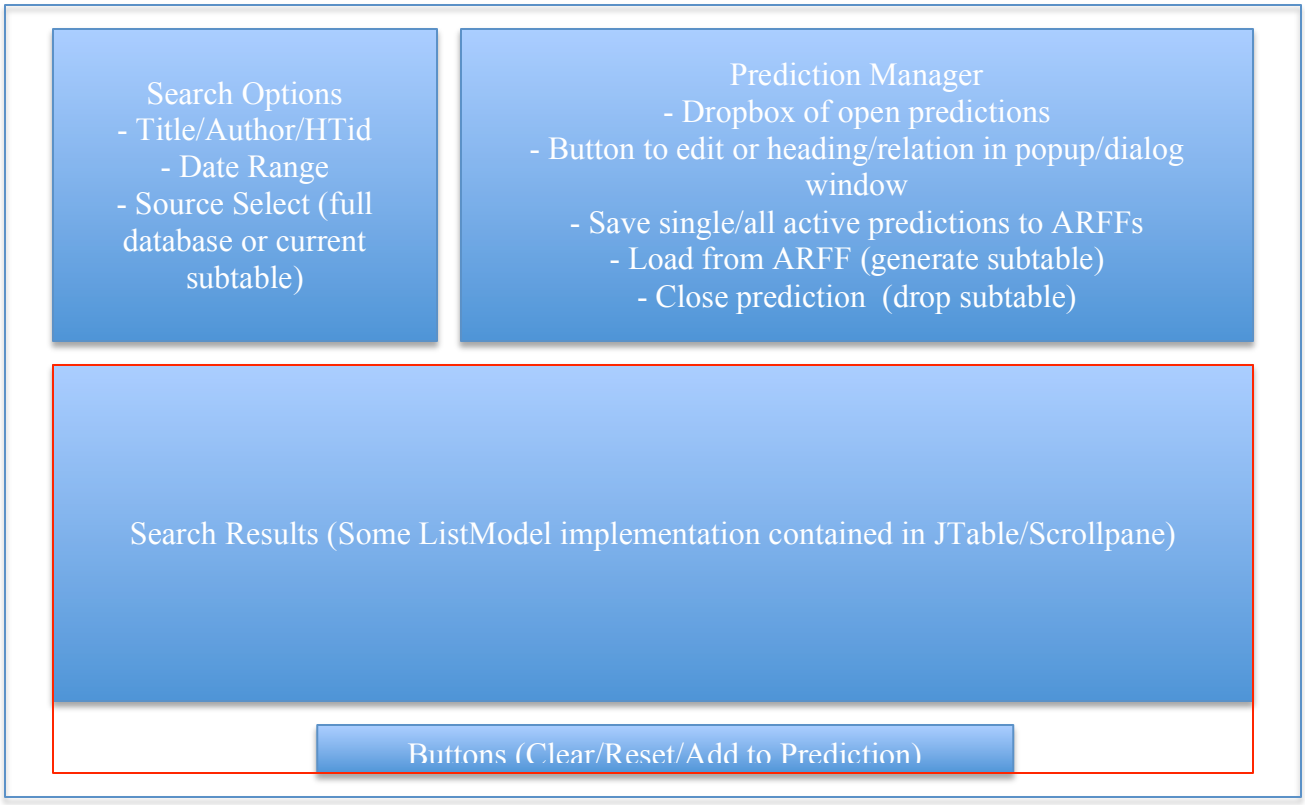
Draft Class Descriptions for Browser UI – Mike B, 6/10/13



Notes:
These are the main components I think are necessary to carry out the proposed example use case. Right now I've kept them somewhat separate based on how I would imagine them fitting into Java's Layout Managers, just sharing by reference certain backend objects when necessary.

Main Class:

- Description: On load, it will set basic UI variables ("exit on close", etc), check for prefs file from past session, initialize DerbyDB object, initialize the four primary GUI classes, create a new window (JFrame), add GUI classes to window
- If no prefs file was found, it will display an error message and then display dialogs to guide users through selecting an existing Derby database or creating a new one using a seed table (metatable.txt or similar file)
- Objects
    - DerbyDB object
    - Search Results
    - (Manual) Search Options
    - Prediction Editor
    - Sub-table Selection
- On close: save current prefs setup

Search Results [Boxed in Red]

- Description: Shows records retrieved from searches and allows users to sort/select results. Buttons beneath will also pass selections to active prediction object. At minimum should show HTid, Author, Title, and Date initially. I think it would also be useful to show a column with prediction assignment, although this could get problematic if items are assigned to multiple open predictions.
- Objects:
    - JTable/Scrollpane (UI classes, contains ListModel & Listeners)
    - Buttons to pass commands to DerbyDB and the Prediction Manager's ListModel
    - Possibly filter buttons to hide all results already in a prediction, just the active prediction, that aren't in a prediction, or that aren't in the active prediction
- References:
    - Active Prediction object: Used to add/remove items from the Results ListModel to the active prediction.
    - Results ListModel: This could live here, but because this object will just display the results and handle selection and passing selected items to predictions (via GUI Table and Button objects), it should probably live in the Search Options unit.

(Manual) Search Options

- Description: Primary GUI interface for Derby queries. Forms will provide search options, and buttons will handle parsing of form data into SQL queries. Query results will be stored in a ListModel (non-GUI backend for displayed list fields).
- Objects:
    - Series of forms/labels for user to decide search options (title, author, htid, date range, prediction proximity [subtable search only])
    - ComboBox to specify search type (default is database wide, entries are added when new subtable ARFF sources are added via Source Manager).
        - Contains OpenPredictions ComboModel. This will be updated elsewhere
    - Search button: ActionListener calls a method that parses the information from the forms into a SQL query string and passes that information to DerbyDB's query method. DerbyDB's results are then passed to Search Results ListModel
    - Clear button: resets all forms to empty or default state
- References:
    - DerbyDB object (passed in from main class)

Prediction Manager

- Description: This manages file i/o for the prediction files and handles SQL subtable generation when predictions are loaded, closed, or created from scratch.
- Objects:

    - Map<String,ARFF> object: This will be where the predictions live in memory. Works like a Python dictionary, so the key of Strings can be retrieved to get a list of all open predictions
    - Dropbox selector that will have a list of open predictions. The currently selected prediction in this object will be considered the "active" prediction by other objects in the application.
    - Buttons to handle the viewing or editing of the ARFF metadata (header, relation and possibly attribute, although I would assume those would fixed around a particular set for this project, so maybe leave list of possible attributes hard-coded).
    - Buttons to handle saving/loading/closing prediction files. In addition to loading the file into an ARFF object, they would also send the appropriate SQL commands to DerbyDB to generate a subtable (or drop the table, if closed).
    - A ListModel/JList/ScrollPane stack just for internal use that displays the contents of the "active" prediction
- References:
    - DerbyDB (from main class): used just to generate or drop subtables when predictions are loaded or closed
    - Open Predictions ComboModel (updated whenever a prediction is open or closed).