Ellen Louie

FemFinder – Component and Interface Design

Data Structures:

- Feed: A feed object represents the RSS feed that a user has chosen to filter.
    - It has the fields title, link, description, copyright, date published, and a list of articles.
    - The list of articles in the particular RSS feed will be represented as an ArrayList of article objects.
- Article: Each article is an object with the fields title, description, link, and author.

External APIs:

- Newspaper: A Python library for article scraping and curation.
    - This library will be used to extract the plain text of articles from the HTML of the webpage they came from so they can be analyzed by a machine-learning algorithm.
    - API Documentation: http://newspaper.readthedocs.org/en/latest/
- Weka: A Java library of machine-learning algorithms.
    - Weka will process the plain text of articles to classify whether the article is related to women or not.
    - API Documentation: http://weka.sourceforge.net/doc.stable/

Components:

The components of my desktop application include (1) a machine-learning algorithm, (2) a local database, (3) a nonprofit organization finder, and (4) a user interface.

(1) For the machine-learning algorithm I will be using a Naive Bayes classifier from Weka. As mentioned previously, this algorithm will be run on the plain text of an article to determine whether it is related to women or not. If it is, the article will be added to the Feed ArrayList of articles and displayed to the user. Pseudocode for analyzing articles using the Naive Bayes classifier is below.

```
analyzeArticles(feed)

    create new ArrayList to hold positive instances

    for i=0 to numArticles

        create word vector from article

        use previously trained Naive Bayes classifier
```

```
            if positive instance

                add to new ArrayList

        return ArrayList
```

(2) The local database will be a MySQL database. Since FemFinder is a Java desktop application, it will use the Java Database Connectivity (JDBC) API to connect to the database. The documentation for JDBC can be found here: http://docs.oracle.com/javase/7/docs/technotes/guides/jdbc/. The main purpose of having a database is to store information about the nonprofit organizations, specifically, a nonprofit's name, donation link, and keywords relating to its issue area. The keywords will be used by component (3) to find organizations that relate to a particular issue area of an article. Pseudocode for querying the database using JDBC is below.

```
queryDatabase(keyword1, keyword2, keyword3)

    load the MySQL driver

    open a connection to database

    execute a query

    extract data from result set

    close all database resources

    return list of nonprofit organizations
```

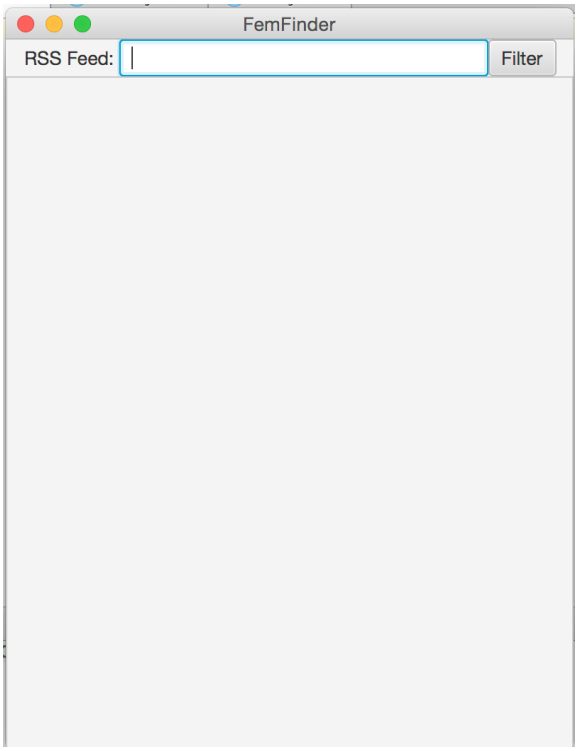
(3) The nonprofit organization finder will use keywords from an article, which will be extracted using Newspaper's natural language processing (NLP) function, to find a nonprofit that matches the issue area of an article. To accomplish this task, the database will need to be queried to find organizations that have similar keywords to the articles. Pseudocode to find nonprofits is below.

```
findNonprofits(article)

    call Newspaper NLP python script

    queryDatabase(keyword1, keyword2, keyword3)

    if matching nonprofit is found
```

return “Would you like to donate to one of these organizations?”

else return “Sorry, no organizations were found”

(4) The user interface of FemFinder will be programmed using JavaFX, which is Java’s replacement for the Swing GUI library. The documentation for JavaFX can be found here: http://docs.oracle.com/javase/8/javase-clienttechnologies.htm. It will be a fairly simple user interface with only one main screen, which is what users will see when they open the application. A screenshot of this is displayed to the left. A user can input the link to an RSS feed in the text area and click the ‘Filter’ button. This will trigger a Newspaper python script that will extract the plain text, which will be fed into component (1). This machine-learning algorithm will filter out any articles that are unrelated to women, and the remaining ones, now in a Feed ArrayList, will be displayed in the area below the input box. This area will be a scrollable pane.



To access an article, users can click on the link provided, and it will open in their default browser. Next to each article will be a ‘Donate’ button, which, when pressed, will trigger component (3) that in turn ends up calling component (4). When the database gets queried, it will return a list of a couple nonprofits that deal with similar issues as the article’s content. In the rare case that no matching nonprofit can be found, a message will return saying that it could not find any organizations at that time. This whole process is repeated any time a user inputs a new RSS feed into the application.