Mo Moulton
CSciE10b Final Project
May 2015
TF: David Habermehl

The GeoText Analyzer

Instructions:

To run the GeoText Analyzer, make the GeoTextAnalyzer directory your working directory and type "java GeoText" at the prompt to start the program.

An initial window will display, prompting you to browse for a text file and enter its title and date of publication. Do this. You can get started with the text files provided, or analyze one of your own. Then, hit "Analyze." A new window will appear that contains the preliminary analysis of the text as well as numerous options for customizing your analysis and saving it as a comma-separated variable file in your working directory.

If you want to analyze more than one text, you may click the initial window again and browse for another file. The program can run more than one analysis window at a time.

Description & Features:

The ability to analyze text quantitatively and create graphical representations of the results is becoming crucial in a wide variety of fields, including journalism, media, publishing, and the humanities and social sciences. With the advent of digitized texts in archives and resources such as Project Gutenberg, users have access to a large amount of textual data. Using the GeoText Analyzer, the user can analyze the frequency of the appearance of words in a text or a group of texts quickly and easily. It has several key features:

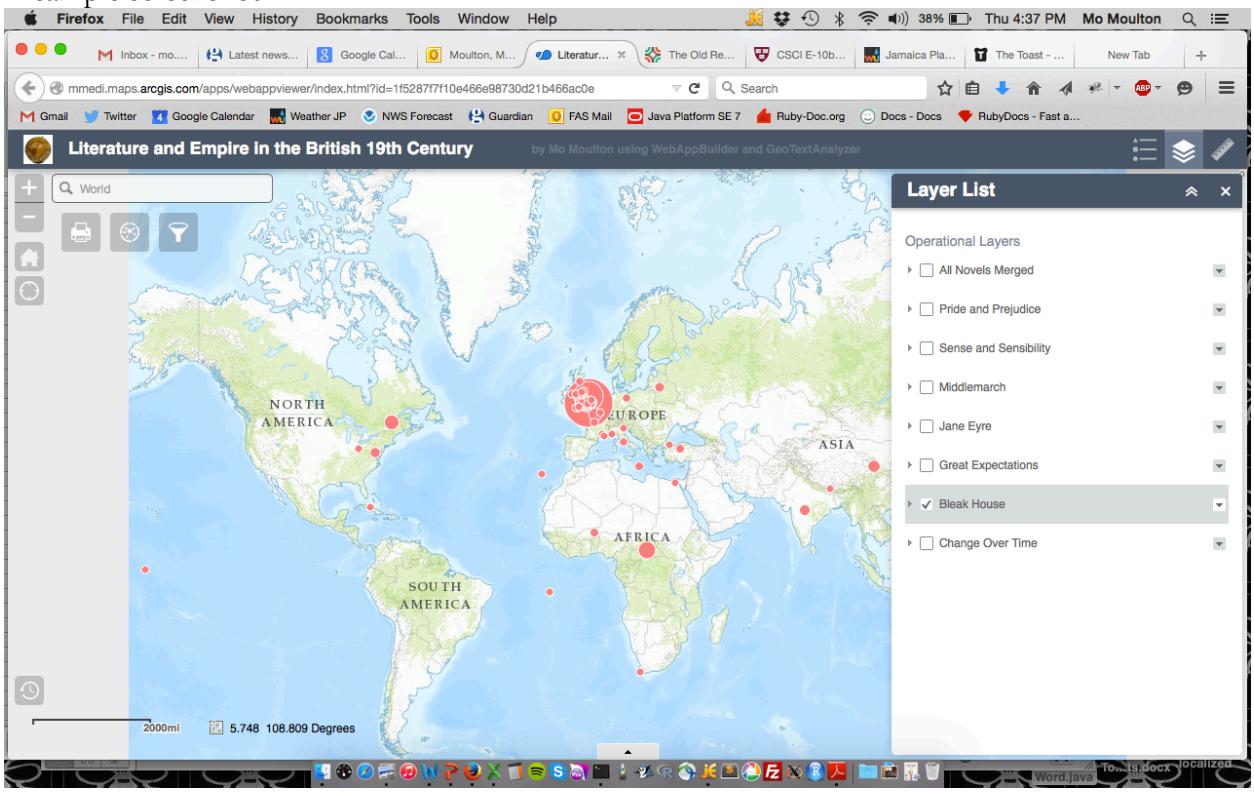
- Customized dictionaries of words and place names. Users can create tailored analyses by generating their own dictionaries of key words or place names. The program calculates the frequency of each word or place name in the text. The program relies on Word and Place classes to save this data in an accessible format.
- Ability to save data in CSV format. CSV (or comma-separated variable) files can be downloaded to facilitate further work, such as the creation of charts, maps, and other forms of data visualization. The Place Name analysis, in particular, produces a file that records latitude and longitude in a way that facilitates the creation of maps through programs such as ArcGIS.
- Text processing (still in prototype). The program currently processes the text so that punctuation is ignored, recognizing that hyphens are used in English prose to separate distinct words. Further development would address internal apostrophes and other similar issues; however, even in its current implementation, the program provides a very reasonable account of standard English prose.
- Generation of a list of proper nouns (still in prototype). The program generates a rough list of proper nouns that, while still overly inclusive in this implementation, nonetheless provides the user with a helpful starting point for identifying place names, in particular.

Demonstration of Place Name Application

For an example of a mapping application created using Place Name data produced by the GeoText Analyzer, go to :
http://mmedi.maps.arcgis.com/apps/webappviewer/index.html?id=1f5287f7f10e466e98730d21b466ac0e

This application, which I created as my final project in CSE8, presents the frequency of geographical place names mentioned in six canonical 19th century English novels.

A sample screenshot:



Expansions

The GeoText Analyzer could be expanded to do a much wider range of textual analysis on the Word and Place classes. For example, methods could be developed to allow the user to graph the appearance of a given word over the course of the text or to calculate its correlation with other words (does a word frequently appear within 100 words of another word, for example).

As suggested above, the text processing could also be improved to deal more thoroughly with the variations of punctuation present in English prose.

Reactions

Working on this project was a great experience. It was really helpful to apply Java programming skills to a problem from my main field (history). I feel able to make small-to-medium programs now to solve other problems in my field, which is terrific after a relatively short time studying Java.

I was surprised by how much time I spent trying to make the program look visually okay. I also spent a lot of time trying out different data structures to store the Word and Place Name objects. I ultimately decided on ArrayLists, because I am familiar with them, though it's possible that in some cases hashes or other set structures would have made more sense. Analyzing large texts (such as *Bleak House* or *Middlemarch*, which have more than 300,000 words) takes a while. The delay is all in launching the analysis window, so I think it's ok for the user, but in the future I hope to learn more about data structures to help me optimize this.

Two suggestions for the future. It might be helpful to have a draft process, to get feedback on a working version and get the experience of revising a project. Also, I would love to know how to let other people use my program without having to compile & run from the command line—maybe a handout or resources on how to put it on the web could be distributed, for folks who are interested.