Search Development Group

Preamble

We've already got our search engine running, but we need your help to make it better. This is definitely one of the more technically-oriented working groups. We're seeking to restructure our search algorithm, perhaps using Elastic Search, so that it provides flexible, real-time results.

Main Goal

The primary goal of this group is one of the two main objectives for the entire hackathon, to restructure our search algorithm and pre-process data in order to achieve real-time results that provide users with various statistical options to sort their output files.

As a group you will decide on which platform and coding language make the most sense to use (perhaps Elasticsearch or some relational-database structure). We aim to provide results that are as similar to the snippet output files in the github repo for this group as possible. Representing the statistics for each word is essential.

At a minimum, we'd like to produce results that provide term frequency and TFIDF, by both aggregate concept (that is all the terms together), and for each single term. We would also be like to find a statistical measure that will represent the robustness of each passage's representation of the concept in question (that is, a passage with many hits for one term is less preferable than one that provides a dispersed signal composed of many different terms.)

Other goals / instructions (in order of priority)

1. Preprocess as much data as possible, esp. time-consuming calculations;
2. Our old R code includes no ability to identify most-frequent-words from loading passages; consult with Web Development group to develop this function;
3. Identify statistical means of measuring robustness of concept signal;
4. Provide the ability to select metadata categories for inclusion;
5. Provide the ability to define a low-pass filter (that is, a cut-off for results – i.e. term frequency > n);
6. Provide the ability to quickly define filters based on authorGroup, titleGroup, etc.
7. Extract txt files from HathiTrust corpus (in a separate AWS instance)
8. Develop EEBO corpus for search (in a separate AWS instance)
9. Develop HathiTrust corpus for search (in a separate AWS instance)
10. Work with Web Development group to combine search engine with website's back end.
11. Experiment with using word2vec to analyze corpus and passages;
12. Integrate analytic tools based on suggestions of DataVis group

GitHub Contents (may be added to before and during the event):

- Snippet output files (to be used as models)
- Earlier iterations of the code we have used for a) dividing each volume into passages, and b) running a search. Both are written in R.
- Group Notebook
- Code Uploads
- You may want to consult the King article in the "Background Reading" folder. It contains a description of a similar search algorithm on p. 12

Why R Won't Work

The proof-of-concept search engine was completed in R. Unfortunately, R requires that the data be loaded into large document-term matrixes, which must be loaded into working memory each time a search is performed. This process takes far too long (~1 hour for a single search!)

Other groups' tasks that might be of interest

- **Search Development**: Liaise with group to discuss data structure;
- **Corpus Development**: They'll likely need help with coding, whether that's to scrape online data repositories or to associate ECCO metadata with other named entities.
- **DataVis and Analytic Tools**: The output files produced by your search algorithm will be the basis of their visualizations.
- **Research Development**: These participants are likely useful to consult to discern what output data is most useful.