Report on Serendip

URL: http://vep.cs.wisc.edu/serendip/

Requirements: Python, Mallet

Introduction

Serendip provides rich visualization capabilities for topic models, along with methods of sorting, filtering, and annotating topic model data. Serendip is unusual in providing three metrics for ranking relationships between topics and documents:

- Frequency (the percentage of a given topic accounted for each word) -- biased towards words appearing in many topics
- Information Gain (the information words gain towards identifying a given topic) -- biased towards rare words that best distinguish topics
- Saliency (frequency multiplied by information gain) -- finds salient words across an entire model, not just within a topic

Saliency is the default ranking metric.

Technology Overview

Serendip runs in a Python-Flask environment. It comes with a separate command-line tool to call Mallet and generate topic models. The Mallet output data is deposited in a Corpora folder and can then be accessed by the Serendip interface. In addition to implementing Mallet, the command-line tool generates multiple files used by the interface to navigate and manipulate the data. Therefore, Serendip will not work if independently generated Mallet output files are deposited in the Corpora folder. It is possible that the script could be modified to read independently generated Mallet data, but this would require some hacking of the Python script.

Serendip is very difficult to install. If one's Python environment is Anaconda, following the website's instructions places Serendip in the `Anaconda/Lib/site-packages/VEP_Core-1.01-py2.7.egg` folder. It is also necessary to install the raven module (`pip install raven`). Although it is possible to provide the command-line tool with a path to input data, I found that it did not work unless I placed my data in the Corpora folder. This is inconvenient because it had to be moved to that deeply embedded path inside the Anaconda folder. The tool *did* read my stop words file from my WE1S-workspace location. However, I had to hack the script in order to give it an acceptable path to call Mallet. I also had to download gzip.exe for Windows and hack the script to give it the path to that application.

After these modifications, the main Serendip screen works, but the Text Viewer does not because one of its functions calls the wrong Flask route. Flask's suggested correction works, but sever other Serendip

functions seem not to generate output, and this may be the result of further code bugs which I have not yet identified.

For my experimental topic model (25 topics and 3312 documents), the command-line tool took 30 minutes to generate the topic model and associated files for this corpus. The Mallet run took five minutes, and the rest of the time consisted of "tagging texts". Because the Serendip interface runs in the browser, it is subject to certain limitations. In my experiments using a model, many functions that needed a screen re-draw caused the browser to hang for approximately 5-15 seconds. Hence, exploring a topic model is not always a quick process.

Because of the small, but for some challenging, difficulties in setting up Serendip, it might be ideal to install it on some lab computers. A Python script with a separate config file might allow users to generate new topic models and place the output in the right locations, without the user's having to enter confusing command-line arguments.

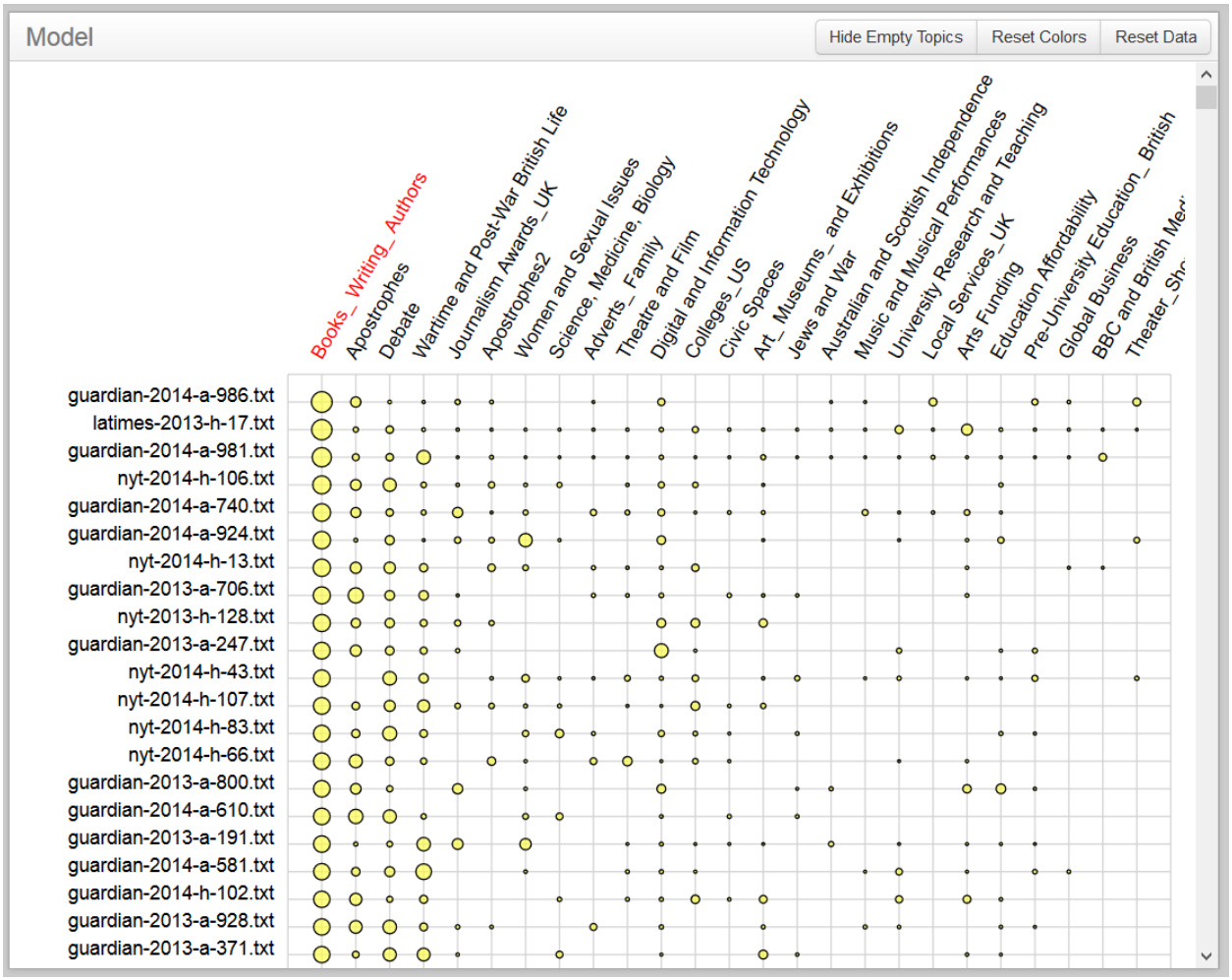
Screen Shots

The following screen shots illustrate the major functions of Serendip. All screen shots show the same 25-topic model of *The Guardian*, *The LA Times*, and *The New York Times* from 2013-2014, using the query terms "arts", "humanities", and "liberal_arts". Since the was a separate run from my earlier 25-topic model, done without random seed and with a more updated stop words list, the topics differ somewhat from the earlier model. However, I have labeled topics as closely as I dared to those of the earlier model so that the two can be roughly compared.

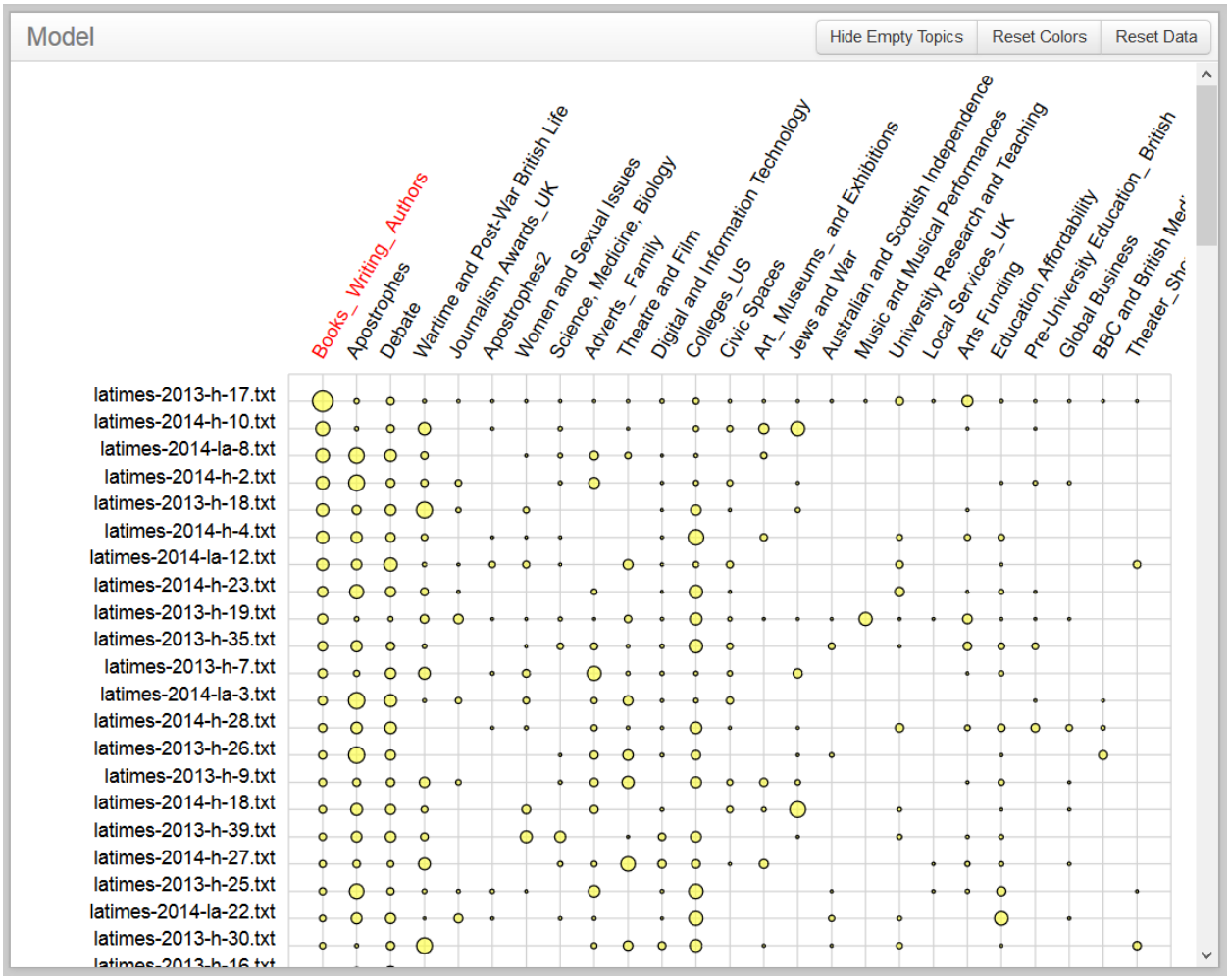
Overview of the Serendip Workspace

Close-Up of Model Viewer

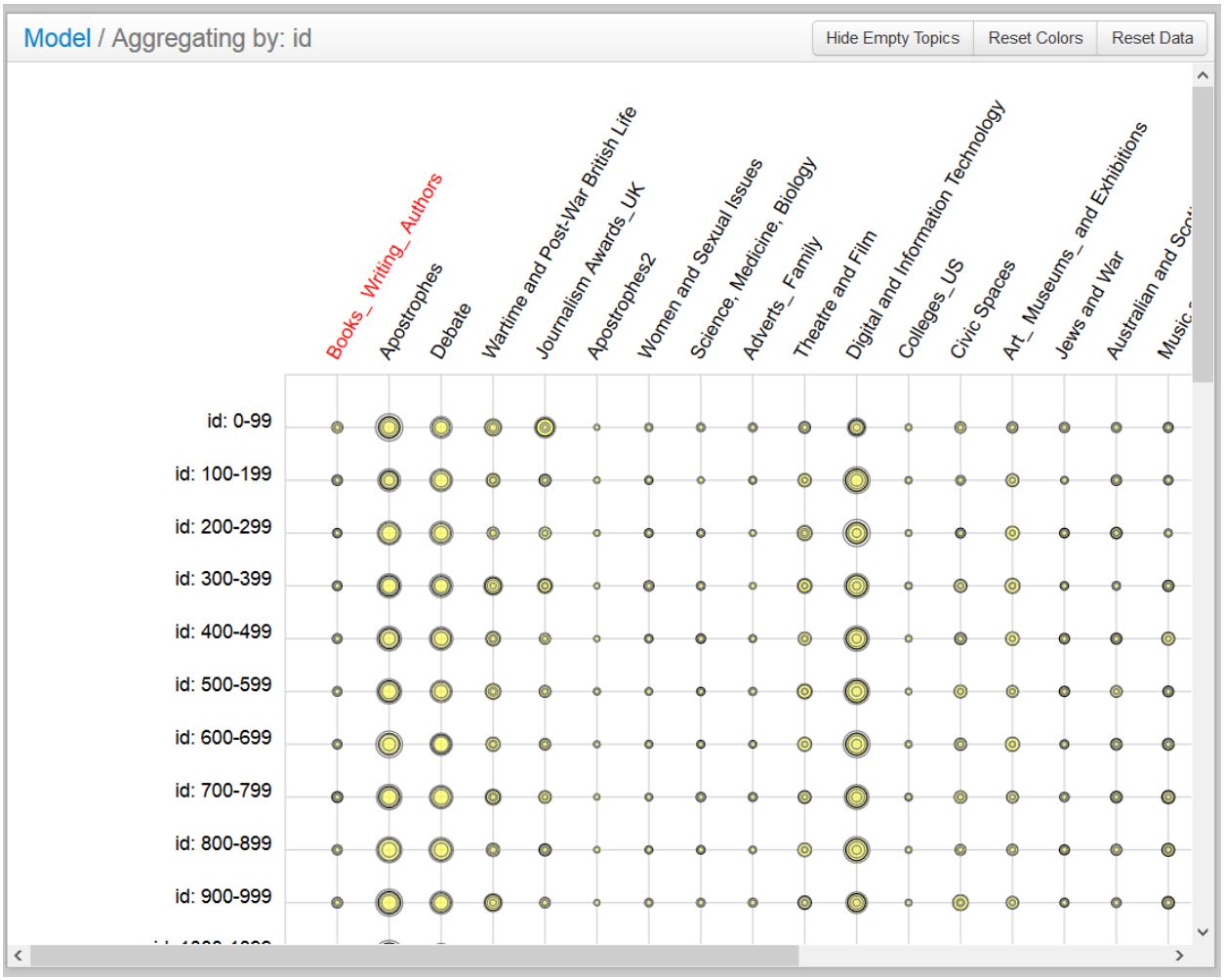
Topics are sorted by similarity to average results of the **Books_Writing_Authors** topic.



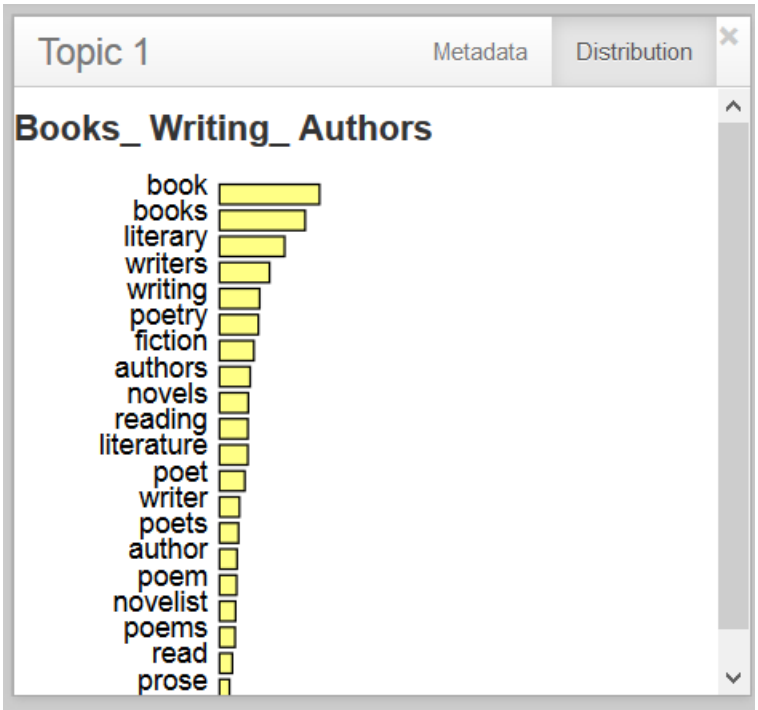
Showing Just the *LA Times*



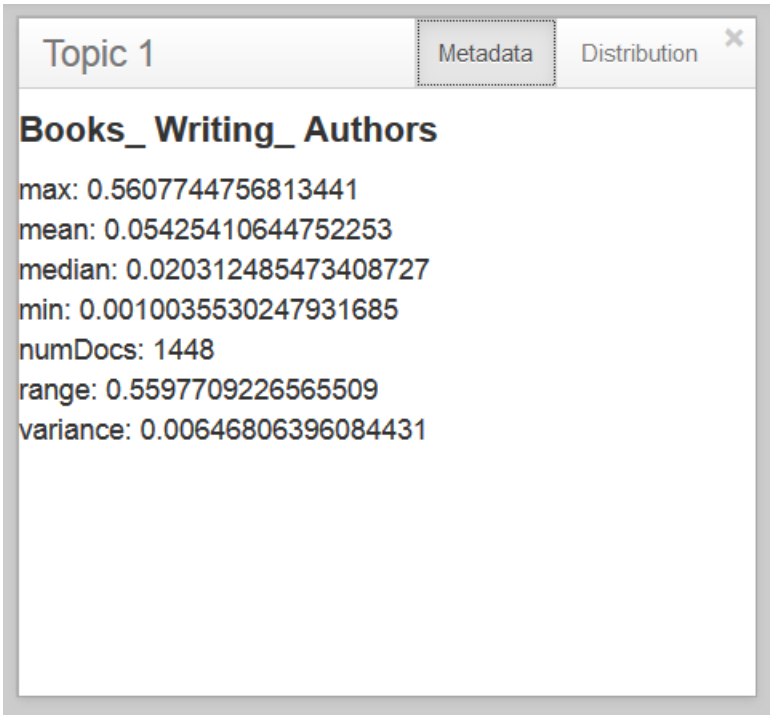
Showing Aggregated Data for *LA Times* in Chunks of 100 Documents



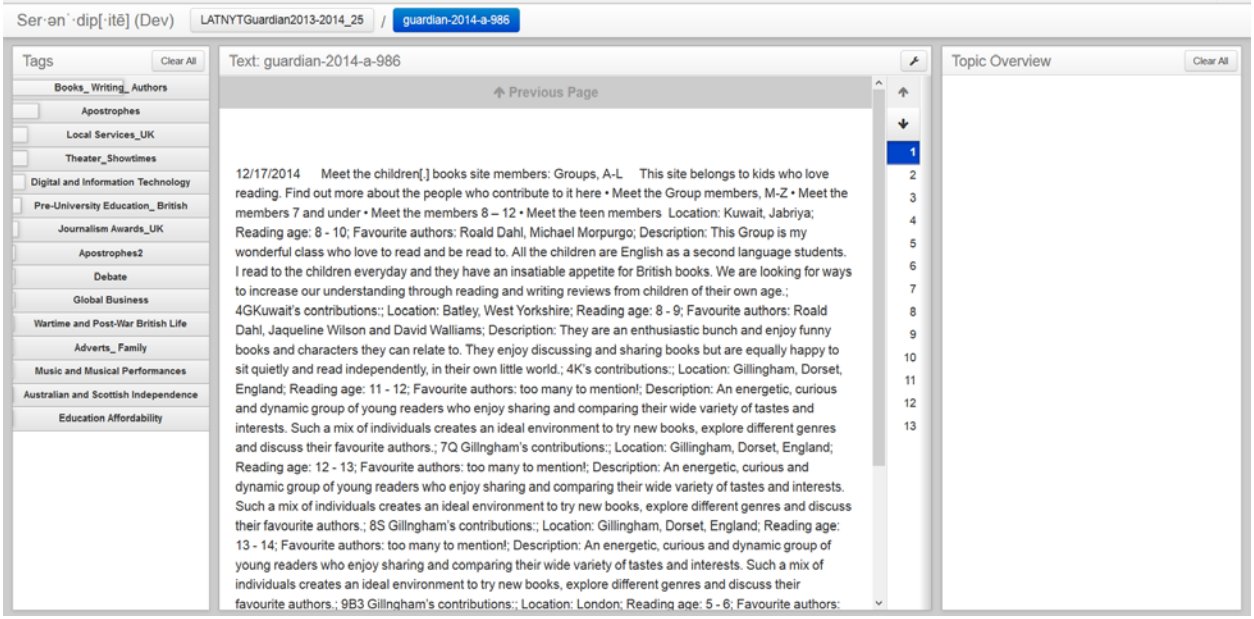
Term Distribution for Books_Writing_Authors



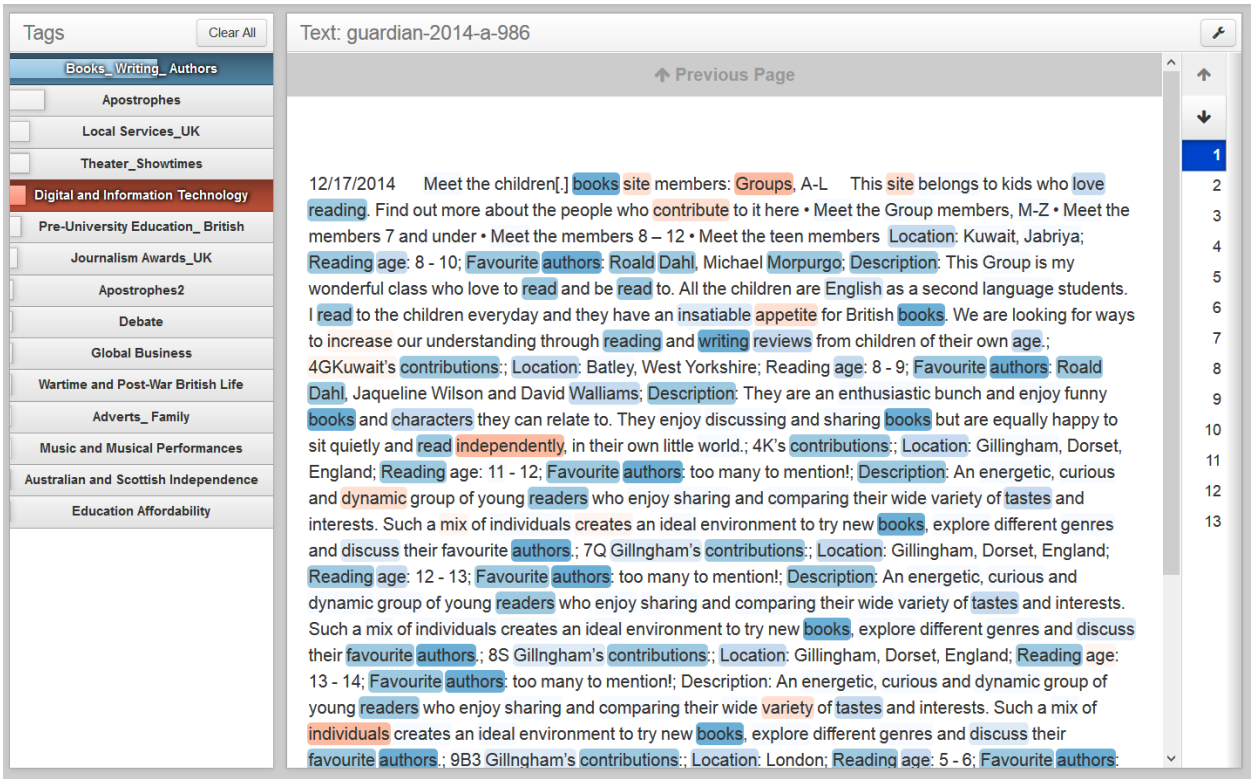
Metadata for Books_Writing_Authors



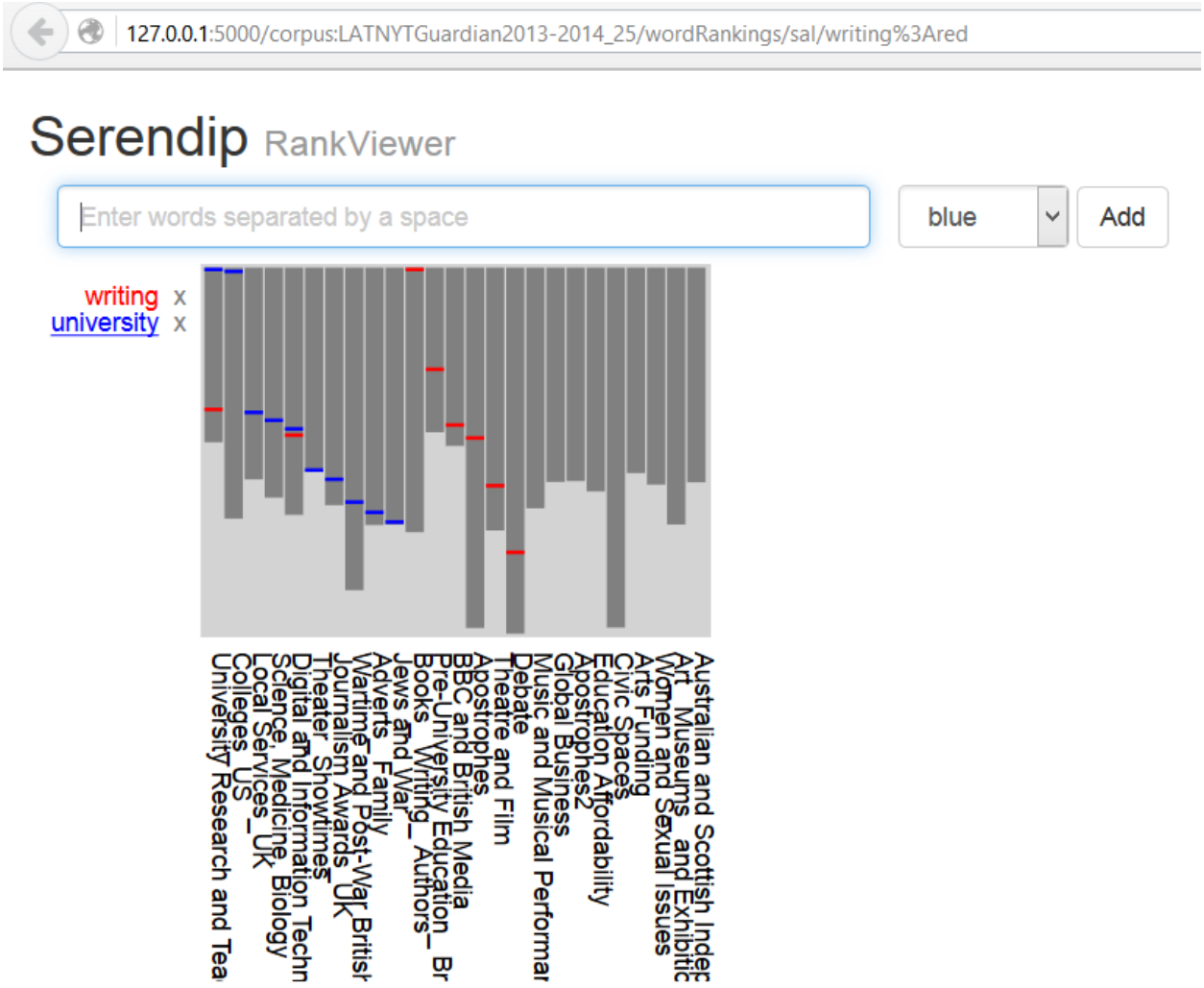
TextViewer Overview (Showing Guardian-2014-a-986.txt)



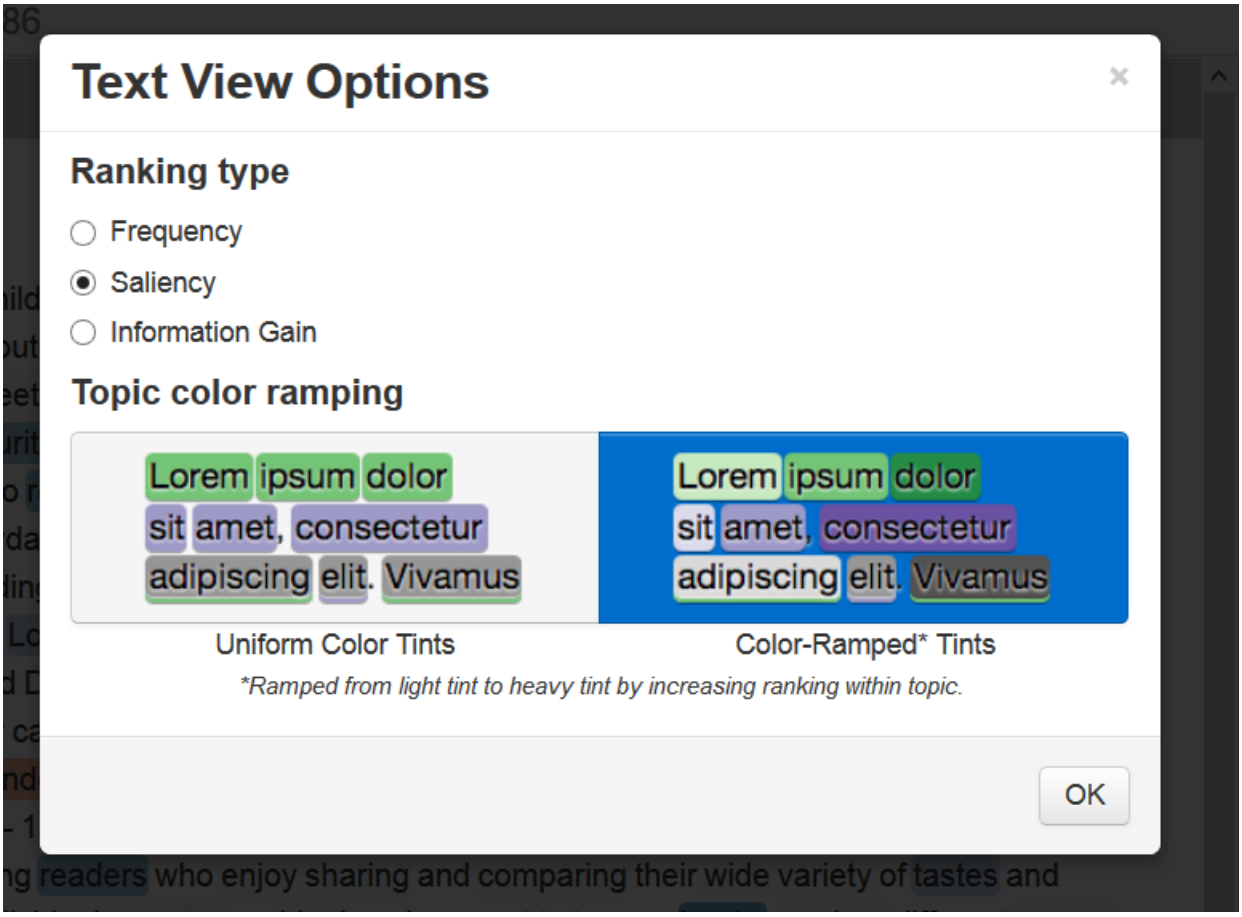
Vocabulary from Books_Writing_Authors and Digital and Information Technology Topics



Rank Viewer (Showing word frequency rank per topic in Guardian-2014-a-986.txt)



Ranking Options



Word Rankings for the Entire Corpus