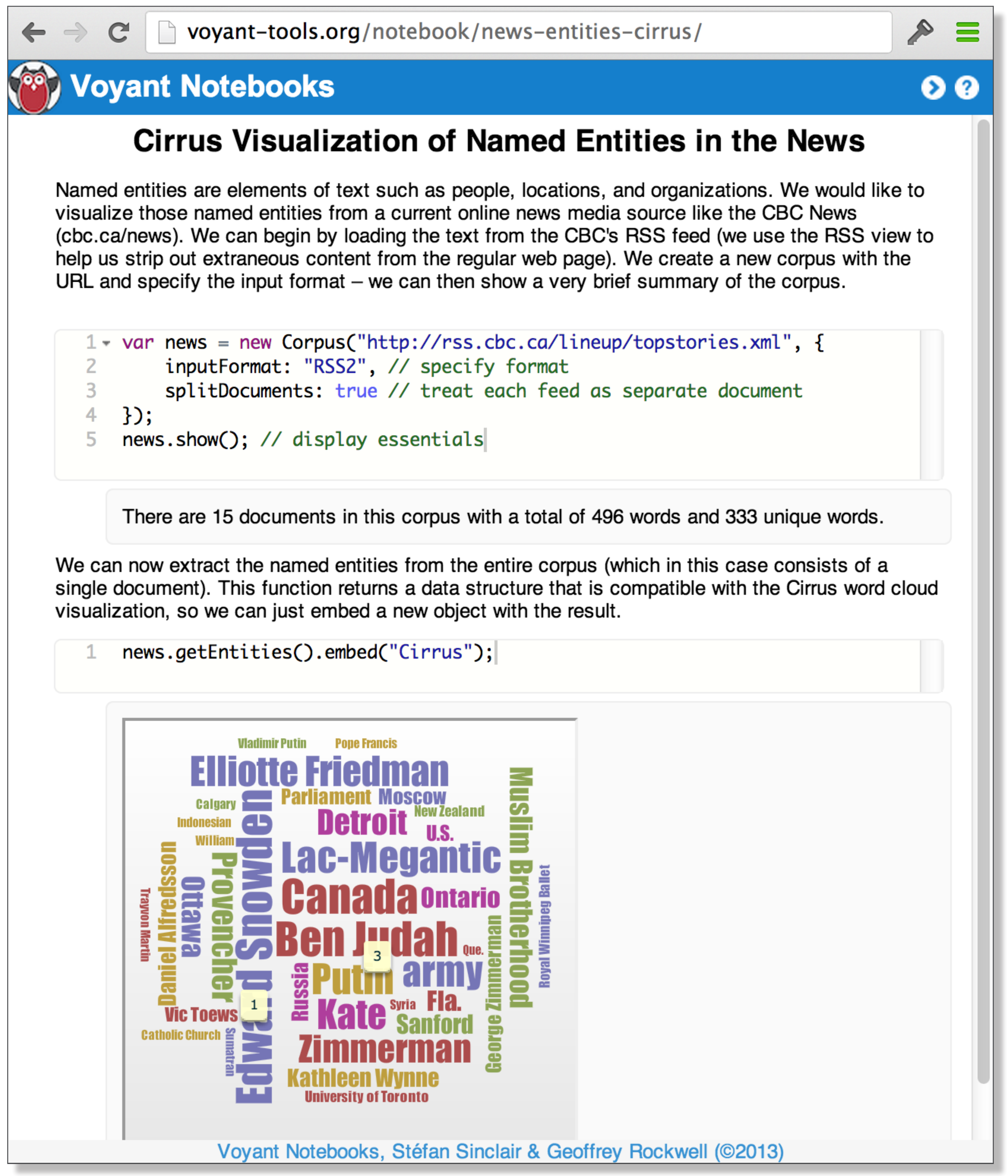
# Voyant Notebooks

## Literate Programming, Programming Literacy

**Voyant Notebooks is a web-based platform for digital scholarship. It combines the powerful back-end analytic capabilities of Voyant Tools with front-end web-authoring capabilities for text, code, and visualization tools.**

**The Problem:** Conventional humanities scholarship privileges interpretation and argumentation in prosaic style. Some digital humanities work makes use of complex analytic and algorithmic operations. The two modes of writing are very challenging to combine in existing publication platforms (print and digital), especially in ways that document and explain procedures adequately without distracting too much from the arguments.



voyant-tools.org/notebook/news-entities-cirrus/

Voyant Notebooks

### Cirrus Visualization of Named Entities in the News

Named entities are elements of text such as people, locations, and organizations. We would like to visualize those named entities from a current online news media source like the CBC News (cbc.ca/news). We can begin by loading the text from the CBC's RSS feed (we use the RSS view to help us strip out extraneous content from the regular web page). We create a new corpus with the URL and specify the input format – we can then show a very brief summary of the corpus.

```
var news = new Corpus("http://rss.cbc.ca/lineup/topstories.xml", {
    inputFormat: "RSS2", // specify format
    splitDocuments: true // treat each feed as separate document
});
news.show(); // display essentials
```

```
There are 15 documents in this corpus with a total of 496 words and 333 unique words.
```

We can now extract the named entities from the entire corpus (which in this case consists of a single document). This function returns a data structure that is compatible with the Cirrus word cloud visualization, so we can just embed a new object with the result.

```
news.getEntities().embed("Cirrus");
```

Voyant Notebooks, Stéfan Sinclair & Geoffrey Rockwell (©2013)

**Literate Programming**

Some thirty years ago Donald Knuth, a computer scientist, proposed literate programming as a better way of organizing narrative and code (1984). Knuth argued that more emphasis should be placed on explaining to humans what computers are meant to do, rather than simply instructing computers what to do. Knuth was especially interested in weaving together macro-style code snippets with prose that provided a larger narrative context, not merely functional comments of specific lines of code that are the distilled remnants of an intellectual process.

Literate programming has been more influential in theory than in practice (Nørmark), despite several utilities and environments including Mathematica, Knuth's (C)WEB, Sweave for R, and Marginalia for Clojure. Perhaps the exigencies of programming in the real world correspond poorly with the vision of Knuth of the programmer as author: "the practitioner of literate programming can be regarded as an essayist, whose main concern is with exposition and excellence of style" (1992, p.1). However, that balance of essayist and coder strikes us as perfectly appropriate for the digital humanities, a natural blend of the expression of intellectual process with the exposition of technical methodologies. The prose can gloss the code, or vice-versa, in a symbiotic relationship that serves to strengthen an argument and demonstrate its own workings.

One of the most significant potential benefits of the literate programming paradigm is pedagogical: these works can both explain an interpretive insight and present the methodology for reproducing the data or results that were part of the process. Many widely-read digital humanities blogs already present these characteristics of exploration, explanation, interpretation and step-by-step instructions (see for example blogs by Ted Underwood, Benjamin Schmidt, Lisa Rhody and Scott Weingart). Literate programming can be more self-contained and more useful for those learning new methodologies and new programming techniques. This is about the principles of literate programming, but also about the potential for increasing programming literacy.

**Challenges:**

- code execution in an asynchronous architecture,
- avoid browser freezes during longer executions,
- mitigating the security risks of user-defined JS code,
- managing access rights to underlying data,
- variable scoping across editor instances & components,
- embedding of Voyant tool panels and other services,
- flexible API for different programming levels and styles,
- including both client-side and server-side operations,
- efficiency of often executed code snippets.

**Technologies:**

- client: ExtJS, jQuery, CKEditor, Ace, D3.js, etc.
- server: Apache Software Foundation, StanfordNLP, etc.

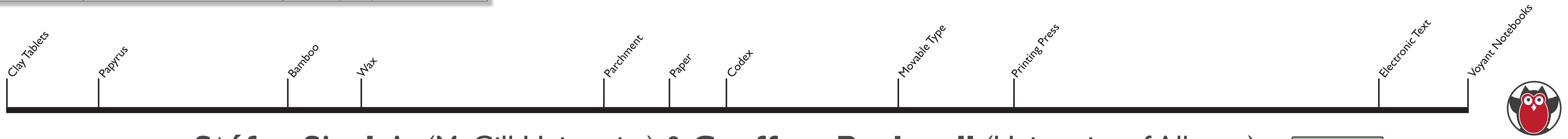
**References:**

- Knuth, Donald. "Literate Programming," The Computer Journal 27, no. 2 (1984): 97-111, 1.
- Knuth, Donald. Literate Programming," Stanford University Center for the Study of Language and Information, 1992.
- Nørmark, Kurt. "Literate Programming: Issues and Problems," (last modified August 13, 1998), bit.ly/10IVFCx
- Sinclair, Stéfan and Geoffrey Rockwell. "Teaching Computer-Assisted Text Analysis: Approaches to Learning New Methodologies" in Digital Humanities Pedagogy. Open Book Publishers, 2012.

**Thanks:** Matthew Milner, Andrew Macdonald, & more: bit.ly/13FUlh7

Social Sciences and Humanities Research Council of Canada
Conseil de recherches en sciences humaines du Canada
Canada

**A Solution**: Voyant Notebooks is a new digital scholarship platform that's intended to support user-friendly web-based authoring of text, live (executable) code snippets, tool results, and visualizations. In addition to writing text, authors can use the Javascript-based macro language as a glue between the sophisticated front-end functionality of Voyant Tools and its powerful back-end analytics engine. Readers can interact with live output and they also have access to data and code to learn, explore and debate new methodologies.



**Stéfan Sinclair** (McGill University) & **Geoffrey Rockwell** (University of Alberta)