

TextPool: Visualizing Live Text Streams

Conrad Albrecht-Buehler*
Northwestern University

Benjamin Watson*
Northwestern University

David A. Shamma*
Northwestern University

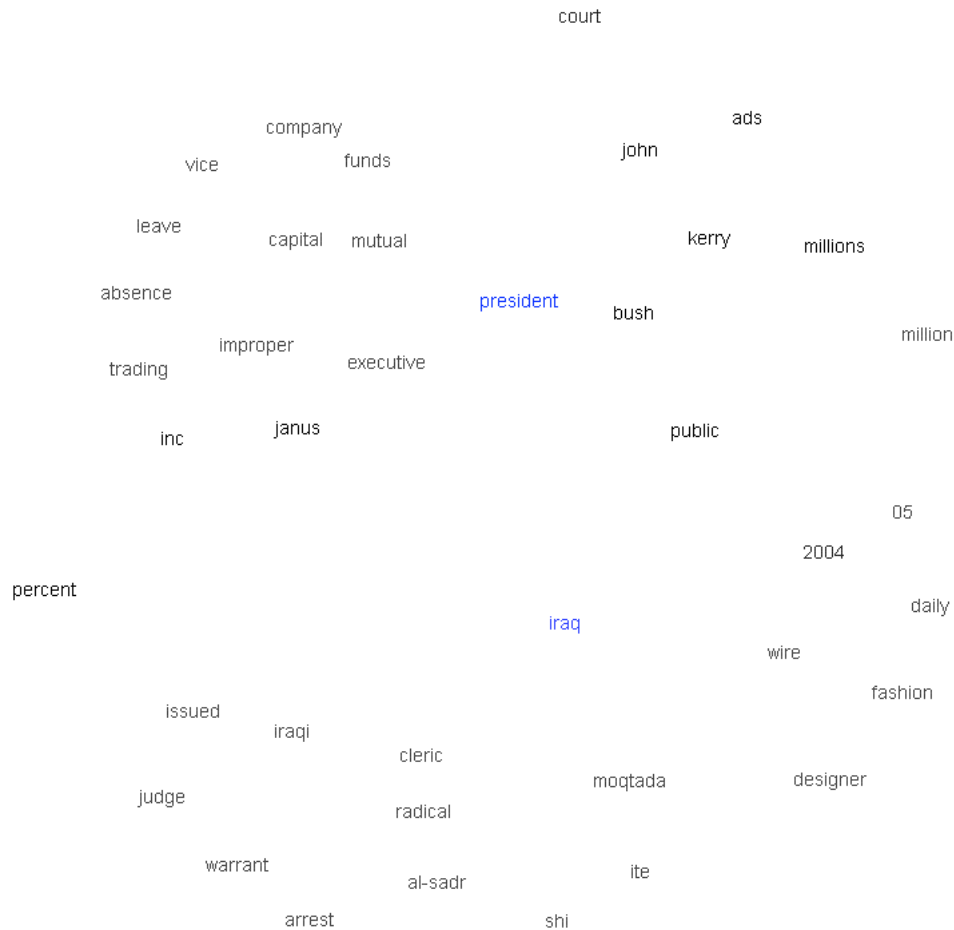


Figure 1: A *TextPool* visualization of six hours of content in several news feeds on Monday, April 5, 2004. Here the user has focused on stories related to the terms “president” and “iraq”. This focused display is derived from 137 stories as represented by 1,662 terms. Note the discussion of improper trading, an Iraqi cleric, and fashion.

ABSTRACT

In today’s fast-paced world, it is becoming increasingly difficult to understand and act promptly upon the content of the many information streams available to us. *TextPool* addresses this problem by quickly summarizing recent content in live text streams, such as newswires and closed captioning. The summarization is a dynamically changing textual collage that clusters related terms. We tested *TextPool* with the content of several RSS newswire feeds, which are updated roughly every five minutes. *TextPool* was able to handle this bandwidth well, and produced useful summarizations of feed content.

CR Categories: I.3.8 [Computing Methodologies]: Computer Graphics – Applications; I.7.0 [Computing Methodologies]: Document and Text Processing – General

Keywords: information visualization, data streams, text layout, information retrieval, newswires.

1 VISUALIZING LIVE TEXT STREAMS

The informational context in which we live and work is becoming increasingly complex. It is certainly richer, with information much more accessible than it ever has been. It is also faster-paced, with news and developments spreading over the media and the internet more quickly than they would have even two decades ago. Such an environment requires new capabilities in information synthesis, enabling analysts and researchers to understand broad trends emerging from many disparate sources. These syntheses

*{conrad,watsonb,ayman}@cs.northwestern.edu

must also be timely, enabling prompt action – in today’s competitive world, the first response to a trend is usually the most effective response.

Yet the same technology that has produced this information complexity offers little help in quickly understanding and acting upon it. Visualization is a useful tool, but its techniques are typically too slow or simply not designed for achieving understanding of highly time-sensitive and dynamic information streams.

TextPool is a tool for visualizing and maintaining a current understanding of live streams of text such as newswires and closed captioned television. *TextPool* buffers and processes these streams in real time using information retrieval (IR) techniques, extracts the most significant terms from the buffered streams, and displays related terms next to one another in a text collage (Figure 1) that is adjusted dynamically in response to user interaction and changes in stream content.

2 PREVIOUS WORK

Research on visualizing text, and particularly text streams, is very limited. Most work to date has focused on visualizing unchanging document collections. The SPIRE [17] and DEPICT [10] systems both transformed documents into feature vectors that were visualized in two or three dimensions using glyphs. Users could then see thematic groups within the document collection. The TOPIC ISLANDS system [5] visualized the contents of an individual document by performing a thematic analysis, applying wavelet filters, and then using any of several largely two dimensional visualizations. The ThemeRiver system [2] was centrally concerned with change in thematic content over time, but its analysis was post-hoc: the collection of documents represented a thematic recording and was not changing during analysis. Users identified certain key themes, and statistics such as frequency of discussion were visualized using a “river”, with colored bands varying in thickness across time.

Wong et al. [18] were directly concerned with visualizing the entirety of a changing document collection using glyphs, and examined several lossy techniques that might speed up their multidimensional scaling (MDS) layout algorithm in such a context. These lossy techniques included reducing the number of elements in document vectors, sub-sampling the document space, and performing an approximate incremental layout of newly arriving documents. Their experiments convinced them that layout speed could be improved without significantly harming output quality. However, they did not actually implement a real time system visualizing a live text stream.

In contrast to these systems, *TextPool* visualizes a constantly changing text stream in real time, adapting its display to stream content and user interaction. *TextPool* achieves this by visualizing only the most recent portion of the stream, rather than the entirety of any document collection or stream recording. *TextPool* is therefore a tool for summarizing and monitoring the current state of one or more streams, not for analyzing any significant portion of stream history. While we are not aware of any similar applications in the information or scientific visualization literature, two artistic visualization systems [12,13] function very similarly.

3 READING LIVE TEXT STREAMS

Our goal was to build a tool for visualizing text streams. To do this we needed to segment streams so that they could be analyzed on a story by story basis. For our test dataset, we chose news stories. Not only are news stories published frequently – in the case of Yahoo! News, as often as every five minutes – but they also represent an aggregate from a broad range of sources.

News stories are first published by various newsgathering organizations, such as wire services, newspapers, or local

affiliates, and are then collected by news syndicators, such as Yahoo! News. These syndicators, in turn, publish the stories as they arrive from the wire services, either as laid out web pages, or as Really Simple Syndication (RSS) feeds. RSS is an XML-based publication format designed to enable source content providers to publish in such a way that web content providers can easily subscribe to, reformat and republish source content [9].

As the syndication feeds change, we monitor and log their content in a database that acts as a text buffer independent of the text visualizer. The visualization client then retrieves the latest feed from the buffer and can request stories over an arbitrary duration.

3.1 Representing News Stories

It is important to remember that an RSS feed is a single document containing one or more news stories. Within the feed, the stories are typically thematically related, such as “Technology News”, “Business”, and “Sports”. As we would like to visualize the exact stories in the news, we segment the feed into individual stories. In doing so, we create a corpus of news stories on which we can easily apply traditional IR techniques to create a vector space model of the stories [9].

Previously Shamma et. al. observed that news stories published via RSS include short (usually 10-30 word) summaries, and that these descriptions are adequate representations of the content of each RSS story [13]. This is because these summaries are typically written by a person very familiar with the story (e.g. the reporter or editor). This allows us to represent each story using a vector of salient terms drawn directly from the summary. We compute this vector by simply removing common stop words such as “and”, “if”, “you” and so on from the summary, and using the remaining terms as the story’s content vector. Because we build content vectors in this way, we do not need to calculate inverse document frequency, which is usually required for IR. For example, if a term such as ‘Iraq’ appears in many stories, it remains in our content vector, instead of being removed as Salton’s method would indicate.

3.2 Salience Across Multiple Stories

The story summaries provide us with content vectors for an individual story, but to further summarize the data to be displayed, we would like a measure of salience across the entire feed. We rely on two properties of term usage within the feeds to estimate this overall salience. First, words that appear together (“co-occur”) likely have related meanings, e.g. “Iraq” and “Saddam”. Second, each wire service collected in a feed will write about topics that they consider important, leading to repetition of co-occurring terms across stories from different wire services. We use this repetition of co-occurring terms as a measure of salience within a news feed. Further, we count these repetitions and use them to enable data space zooming.

4 DYNAMIC VISUALIZATION

We not only sought to present the user with the salient terms from the news feed, but also to convey how those terms are related to one another. To that end, the display that we present is a graph (Figure 2). Nodes represent salient terms from the stream, and are connected if they co-occur within the same news story. The lengths of the connections are scaled by the number of times terms co-occur, so that terms that are closely related and co-occur often are close to another in the graph.

4.1 The Temporal Window

The graph visualizes the recent history of the stream using a user-controlled temporal window. As the stream moves through the window, old news items that have moved beyond the window are removed from the graph, and those that have just been published

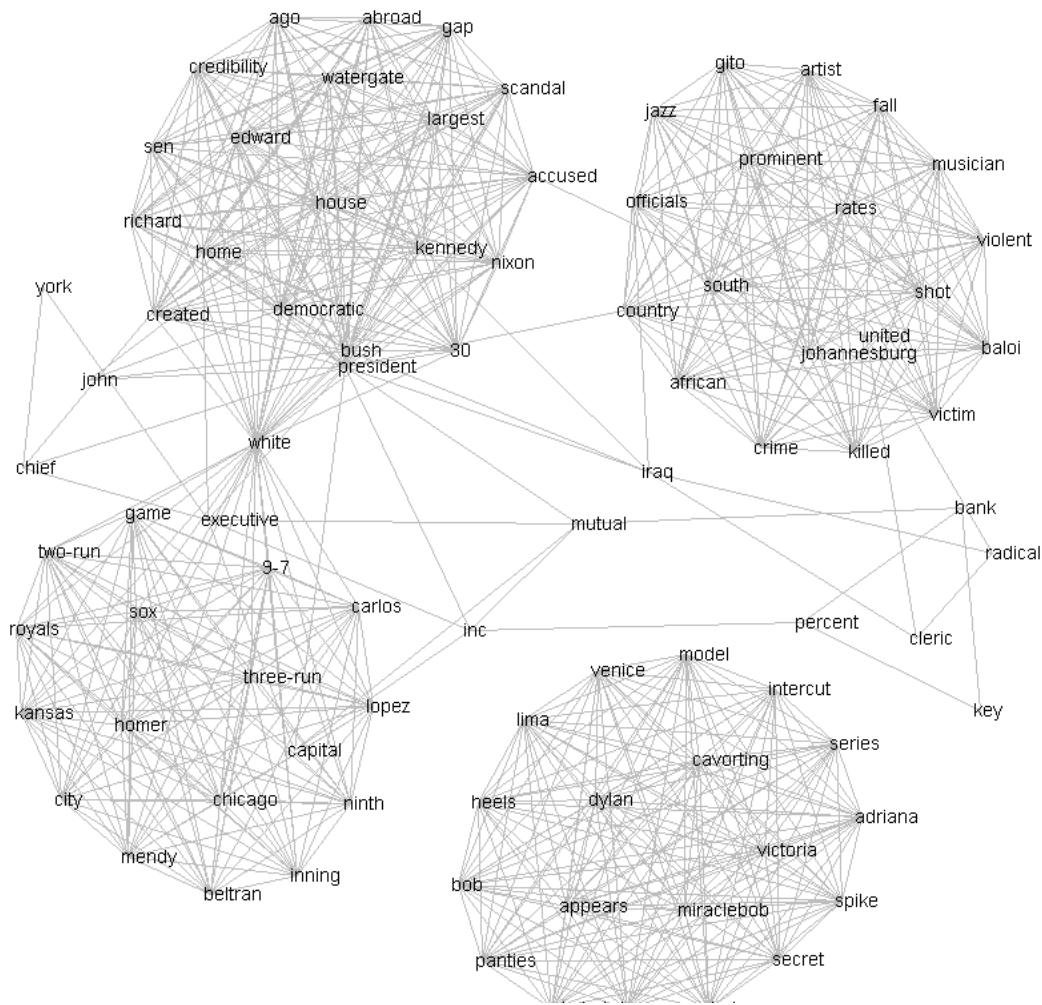


Figure 2: Another *TextPool* visualization. Nodes represent salient terms from the stream, and are connected if they co-occur within a story. Connection length shortens as frequency of co-occurrence increases, revealing term relatedness.

and entered the window are added to the graph. This temporal window allows the user to stay current with the latest information from the stream, while also providing him or her control over the displayed temporal context. This control enables the user to look at the topics discussed in the last 2 hours, the last 2 days, or even the last week.

4.2 Display Elements

Nodes. A key design decision in this display was the choice to visualize the salient terms using text, and not another representation such as glyphs. For our monitoring application, text offered a unique communicative power. This decision creates several additional constraints. Most importantly, occlusion can seriously limit display clarity. Nevertheless we would not want to eliminate occlusion completely – it can be a useful way of maximizing use of limited display space. We therefore prevent occlusion only between a node and its nearest neighbor. We are also cautious with further manipulation of displayed text, using brightness to indicate the recency of term use, and color only as an indicator of user selection.

Attractors. In *TextPool*, a link between nodes indicates that those two terms were used in the same story (they co-occurred). The more they co-occurred, the shorter the link, and the more related the two terms appear.

Spacers. In fact, the graphs displayed by *TextPool* are fully connected. Any pair of nodes representing terms that did not co-occur have a “spacer” link between them that to ensure that they are far apart. The length of spacers is twice the maximum attractor link length.

4.3 Real Time Layout

A unique feature of *TextPool* is our interactive application of force-directed layout methods. Force-directed layout is a physical simulation wherein nodes are modeled as masses, and their links are modeled as springs. Each spring (link) exerts a force proportional to the difference between its current length in the display and its resting length (as indicated by co-occurrence), and each node moves in the direction of the sum of the forces exerted on it. The simulation determines the layout by trying to equalize all the forces within the spring-mass system (graph) (Figure 3). We use force-directed graph layout instead of eigenvector-based multi-dimensional scaling because the spring-mass simulation can be updated interactively, enabling the user to watch as the nodes reorganize and term relatedness changes over time.

The lengths of *TextPool*'s links change over time as the co-occurrence changes. For example, when one of three stories about "elections in Iraq" expires, the co-occurrence of "election" and "iraq" is reduced and the length of the link increases. To make this

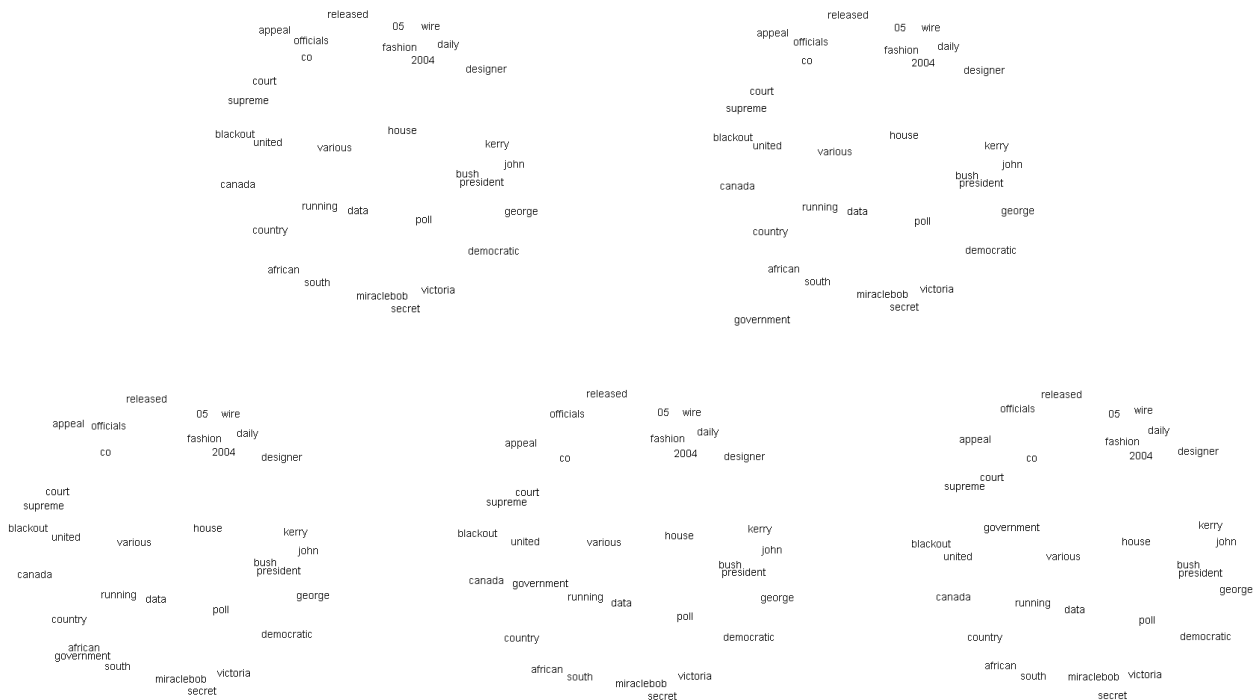


Figure 3: Some new stories arrive in the stream, making the term “government” salient enough for display. The term arrives at the lower left, and gradually works its way into the current visualization, finding an optimal position next to the terms “supreme” and “court”.

motion as compelling as possible, we set the per-node mass so that the resulting motion conveys a familiar sense of momentum and damping.

5 SUPPORTED INTERACTIONS

Users can interact with *TextPool* to change the displayed temporal context, to focus and zoom in on portions of the stream that are particularly interesting, and to highlight term age and relatedness.

Controlling temporal context. With a simple interaction, *TextPool* users can adjust the size of the temporal window, effectively changing the temporal context in which terms appear. For example in the context of a half day, a story about a murder in South Africa may seem quite unimportant. Yet in a smaller two-hour context, it may suddenly seem quite important.

Focusing and zooming. By selecting several displayed terms, users can limit display to the those terms and any that co-occur with them. *TextPool* then allows users to zoom in on the data-space with a slider that controls the minimum number of times a term must co-occur to be displayed. Thus after focusing on a smaller set of terms, users can display additional related terms by lowering this minimum (Figure 1). *TextPool* also permits display space zooming with a slider that scales the lengths of all links. At any point, *TextPool* can also provide an organized list of URLs pointing to the documents containing the currently displayed terms. This is particularly useful once the user has isolated an especially interesting portion of the stream that needs closer analysis.

Highlighting. With the click of a button, *TextPool* users can highlight term age by dimming older nodes, revealing not only which terms have been the topic of recent discussion, but also which terms are “hot” and regularly mentioned (Figure 1). (Term age is the time since the last reference to the term in the stream). By default, attractor links are not displayed, but *TextPool* users can reveal them, highlighting term relatedness. This is particularly useful when the display is densely populated (Figure 2). Users can

also highlight relatedness through motion by temporarily adding energy to the spring-mass system. In the resulting ripple, related groups of terms move in a coordinated fashion.

6 FUTURE WORK

There is much room for improvement of *TextPool*. In the near term, we will be increasing *TextPool*’s capacity so that it can handle even higher bandwidth streams and create visualizations with larger temporal contexts. This will be an important part of our emerging collaboration with Proctor and Gamble, which receives a stream of 50,000 consumer emails per day.

Although *TextPool* supports two very different patterns of use, one passive and the other active, it has not been specialized for either of those uses. We plan to examine the possibility of such specialization – for example, motion might be much more subdued when the user hasn’t interacted with *TextPool* recently and is using it as a peripheral monitor [4,14].

In the longer term, we will also face more basic scalability challenges. For example, display real estate will always limit our ability to visualize the full range of term relatedness. Our current solution is to visualize only those terms which are most strongly related. However, this does not support awareness and understanding of weaker, but still meaningful relationships. Potential solutions are large format display walls, which raise new interaction issues; targeted visualizations, which accept several seed terms to prime the visualization; large virtual visualizations that are only windowed by the physical display (perhaps coupled with zoomable user interfaces [1,6]); and hierarchical clustering approaches. We may also face difficulties with our use of a spring-mass system, which does not scale well ($O(n^3)$) as the number of nodes in the system grows. Some potential solutions have been described by Morrison et al. [6,7], though their focus is not on the interactive sorts of applications we describe here.

We will also experiment with additional display dimensions such as text size, color and motion, subject to the

understandability concerns raised by graphic design [15,16]. As do Lee et al. [3], we believe the potential of motion in conveying meaning is rich. Data attributes that might be mapped to these dimensions include stream ID, the frequency of each term, and the number of documents containing each term.

7 CONCLUSION

We have presented *TextPool*, an interactive system that summarizes recent content in live text streams. *TextPool*'s visual summary is a dynamically changing textual collage, in which related terms are grouped together. *TextPool* can be used in an ambient mode, offering a peripheral awareness of stream traffic; or an interactive mode, supporting a deeper understanding of stream content with focus and zoom functionality. We tested *TextPool* by using it to visualize the content of several RSS news feeds at one time. *TextPool* was able to handle this volume well, and produced useful summarizations of current RSS feed content.

8 ACKNOWLEDGEMENTS

Our thanks to Kris Hammond, Larry Birnbaum and the InfoLab for their suggestions and advice, and to Proctor and Gamble for their encouragement and interest.

9 REFERENCES

1. B. Bedersen & J. Hollan. 1994. Pad++: a zooming graphical interface for exploring alternate interface physics. Proc. ACM UIST, 17-26.
2. S. Havre, B. Hetzler & L. Nowell. 2000. Proc. IEEE Information Visualization, 115-123.
3. J. Lee, J. Forlizzi & S. Hudson. 2002. The kinetic typography engine: an extensible system for animating expressive text. Proc. ACM UIST, 81-90.
4. J. Mankoff, A. Dey, G. Hsieh, J. Kientz, M. Ames & S. Lederer. 2003. Heuristic evaluation of ambient displays. Proc. ACM CHI, 169-176. 2003.
5. N. Miller, P. Wong, M. Brewster & H. Foote. 1998. TOPIC ISLANDS – a wavelet-based text visualization system. Proc. IEEE Visualization, 189-196.
6. A. Morrison & M. Chalmers. 2003. Improving hybrid MDS with pivot-based searching. Proc. IEEE Information Visualization, 85-90.
7. A. Morrison, G. Ross & M. Chalmers. 2002. A hybrid layout algorithm for sub-quadratic multidimensional scaling. Proc. IEEE Information Visualization, 152-158.
8. K. Perlin & D. Fox. 1993. Pad – an alternative approach to the computer interface. Proc. ACM SIGGRAPH, 57-64.
9. RSS Advisory Board. 2003. RSS 2.0 specification. <http://blogs.law.harvard.edu/tech/rss>.
10. D. Rushall & M. Ilgen. 1996. DEPICT: documents evaluated as pictures. Proc. IEEE Information Visualization, 100-107.
11. G. Salton, A. Wong & C. Yang. 1975. A vector space model for automatic indexing. *Commun. ACM*, 18, 613-620.
12. V. Setlur, D.A. Shamma, K.J. Hammond & Sanjay Sood. 2003. Towards a non-linear narrative construction. Proc. ACM Intelligent User Interfaces, 329.
13. D.A. Shamma, S. Owsley, K. Hammond, S. Bradshaw & J. Budzik. 2004. Network arts: exposing cultural reality. Proc. WWW. To appear.
14. M. Weiser & J.S. Brown. 1996. The coming age of calm technology. <http://www.ubiq.com/hypertext/weiser/acmfuture2endnote.htm>. Revised version of: Designing calm technology. *PowerGrid J.*, v1.01.
15. R. Williams. 2004. *The Non-Designer's Design Book*, Second Edition. Peachpit Press, Berkeley, California.
16. R. Williams. 1998. *The Non-Designer's Type Book*. Peachpit Press, Berkeley, California.
17. J. Wise, J. Thmoas, K. Pennock, D. Lantrip, M. Pottier, A. Schur & V. Crow. 1995. Visualizing the non-visual: spatial analysis and interaction with information from text documents. Proc. IEEE Information Visualization, 51-58.
18. P. Wong, H. Foote, D. Adams, W. Cowley & J. Thomas. 2003. Dynamic visualization of transient data streams. Proc. IEEE Information Visualization, 97-104.