STREAMIT: Dynamic Visualization and Interactive Exploration of Text Streams

Jamal Alsakran*
Kent State University

Yang Chen†
University of North Carolina - Charlotte

Ye Zhao
Kent State University

Jing Yang
University of North Carolina - Charlotte

Dongning Luo
University of North Carolina - Charlotte

ABSTRACT

Text streams demand an effective, interactive, and on-the-fly method to explore the dynamic and massive data sets, and meanwhile extract valuable information for visual analysis. In this paper, we propose such an interactive visualization system that enables users to explore streaming-in text documents without prior knowledge of the data. The system can constantly incorporate incoming documents from a continuous source into existing visualization context, which is "physically" achieved by minimizing a potential energy defined from similarities between documents. Unlike most existing methods, our system uses dynamic keyword vectors to incorporate newly-introduced keywords from data streams. Furthermore, we propose a special keyword importance that makes it possible for users to adjust the similarity on-the-fly, and hence achieve their preferred visual effects in accordance to varying interests, which also helps to identify hot spots and outliers. We optimize the system performance through a similarity grid and with parallel implementation on graphics hardware (GPU), which achieves instantaneous animated visualization even for a very large data collection. Moreover, our system implements a powerful user interface enabling various user interactions for in-depth data analysis. Experiments and case studies are presented to illustrate our dynamic system for text stream exploration.

1 INTRODUCTION

Over the past decades, advanced technologies (e.g. cable TV, mobile phone, and internet) in data generation, storage, and communication have greatly increased the quantity and accessibility of text documents in various areas of human society. Massive documents are generated and published at a significant speed, e.g., from daily, hourly, or minutely emails, messages, webs, broadcasts, and TVs. They have introduced an urgent need for efficient storage, processing, retrieval, and analysis of the explosive text collections. Undoubtedly, computer-based visualization tools are among the most effective approaches.

A stream text collection constantly evolves as new documents are continuously generated and published, e.g., the news items from a daily newspaper or an hourly TV broadcast, as more news being released on hourly basis. These sources differ from the traditional text database in which the quantity and the representation (e.g. keywords or topics) of documents are known in advance. Visual exploration of real text streams is a challenging task. First, text streams continuously evolve. Visualization aids should be provided to users to allow them to trace the temporal evolution of existing topics, monitor emerging new topics, as well as examine the relationships between new topics and existing topics. Second, as in other stream processing approaches, a visualization system should process a text stream on-the-fly without pre-scanning the whole stream or assuming a priori knowledge of it. Third, a visualization system should allow users to interactively change their information seeking focus and preference at any time, which will be immediately reflected in pertinent visual effects. Such interactivity is a decisive factor for a visual analytic system in real applications, since the domain users might not have a priori knowledge of the text streams they are working on. Fourth, a visualization system should scale to the large volumes of text streams and respond to evolution of the streams in real time.

In this paper, we propose a novel text stream visualization system. It is based on a dynamic force-directed simulation into which documents are continuously inserted. Each document is represented as a mass particle that moves inside a 2D visualization domain. A potential energy is defined by pairwise text similarity between documents. Minimizing the total potential energy of the system moves similar document particles closer and drives away dissimilar ones, which are achieved by attractive and repulsive forces between particles. Consequently, an equilibrium state of the particles visually depicts the data clusters and outliers at a particular moment. The system automatically adjusts its visual output with newly injected document particles. The dynamic procedure of this change is critical for reducing change blindness when new patterns emerge. This approach has the following features that addresses the above text stream visualization challenges:

Continual evolvement: This physical model is well-suited to visualize text streams for continuous depiction and analysis of growing document collections, where the dynamic nature is simulated through the live behavior of particles. Text documents enter the system at any time and automatically join clusters of related collections. In the meantime, the particles already inside the system travel continuously under the impact of new particles. The visual structures hence gradually evolve as new documents enter the collections without abrupt changes that break the mental picture users already form with existing documents. Erratic motion of particular particles (e.g., moving from one cluster to another cluster) can be used to identify outliers or significant new trends in text streams. This is advantageous to existing static or time-window based visualization approaches, which depict only stationary data patterns or the sporadic transitions between these patterns.

Dynamic processing: Text documents are typically represented and manipulated through the vector of keywords. Existing methods usually calculate similarity between documents from their constituent keywords. The keyword list is predefined and the similarity is statically computed from them. Instead, to make our system suitable for real world applications dealing with unplanned flowing-in data, we develop dynamic keyword vectors that upgrade adaptively from the incoming documents. Essentially, our system does not require a scan of the whole collection of a prerecorded stream before visualization. The visualization parameters and functions can also be managed with respect to the temporal context.

Interactive exploration: We propose a *Dynamic Keyword Impor-*

*Email: {jalsakra, zhao}@cs.kent.edu
†Email: {ychen61,Jing.Yang,dluo2}@uncc.edu

tance that presents the significance of a keyword at a certain time. It reflects user demand or interest so that similarity is updated on-the-fly, and consequently the visualization artifacts. For instance, keywords with increasing importance will make documents having those keywords aggregate closer. In addition, our system includes a highly interactive user interface for effective exploration.

Scalable optimization: We optimize our method by introducing a similarity grid that spatially divides the visualization domain into rectangular cells. When a new document enters the system, it is placed in a cell with the most similar documents so that it will quickly reach its preferred location during simulation. This arrangement greatly increases the computational speed of the system and thus the visual response time. Moreover, our particle system is inherently parallel, similar to an N-body problem. By accelerating the computation on graphics hardware (GPU), our system has very good scalability to accommodate a very large number of particles (documents).

A fully working prototype that visualizes live text streams in real time, named **STREAMIT**, has been built up upon the force-directed dynamic system. It has the following features:

Dynamic visualization and animation: Continuously incoming text documents are visually presented to users in a dynamic 2D display driven by the force-directed model. The evolution of the visual structures and labels reveals the semantic evolution of the text stream being represented. Users can discover emerging patterns online by monitoring the real-time evolution. They can also examine the temporal evolution of historical data through animations that playback the stream evolvement over time.

Interactions: A set of interactions are provided in STREAMIT. First, users can change the visual structure of the display on-the-fly to emphasize the topics of current interest by manipulating keyword importance. They can also highlight documents or keywords of interest in the dynamic visualization to track the semantic evolution around these documents or keywords. Users can also control the maximum age of documents and the maximum number of documents in the visualization. Second, users can retrieve documents of interest through a variety of interactions, such as rubber band selection, search by example, and search by keywords. The selected documents can be sent to a shoebox through which users can investigate them in full detail when they have time.

Since the similarity grid and hardware acceleration are used, STREAMIT can respond to data evolution and user interactions with immediate visual feedbacks for large text collections.

2 Related Work

Many text visualization systems use similarity-based projection to help users get insights from large text collections. For example, IN-SPIRE [23] uses multidimensional scaling (MDS) to map documents with similar contents close to each other, and thus form ”galaxies” or ”mountains” in the displays. Later, InfoSky [2] exploits hierarchically structured documents at each level with Voronoi diagrams. A point placement approach is proposed in [19] to build a hierarchy of the documents and project them as circles, where a circle size is proportional to the number of children. Examplar-based visualization [5] visualizes extremely large text corpus by probabilistic multidimensional projection with approximation and decomposition. Our approach is different from the above approaches in that it uses a dynamic similarity-based projection system to depict text streams.

There exist many approaches to exploring the temporal trends in archives of text with time stamps. For example, ThemeRiver [10] and LensRiver [8] depict the strength changes of individual keywords in a text collection as currents within a river flowing along a time axis. T-scroll [12] employs a novelty-based clustering algorithm on time-series documents, and the transition of the topics is simply visualized as a screen scroll. Meme-tracking [13] extracts keywords from news corpus to generate themes, and then visualizes these themes at different time steps to indicate the flow of stories. Email archives are visualized to explore conversational relationships among individuals by Themail [22] using interaction histories. Interactive, topic-based analysis tool is provided for generating time-based, visual text summary along time axis [14]. Our approach targets at live text streams rather than time-stamped existing archives.

Related to our aim to handle continuous incoming text streams, TextPool [1] produces a visual summary that clusters related terms as a dynamic textual collage. Unlike our method, it visualizes very recent stream content as a partially connected graph, which is “not for analyzing any significant portion of stream history”. Besides, the graph represents salient terms of the stream instead of the documents which limits the visual connections to those terms. Wong et al. [24] dynamically visualize stream data in a moving time window using incremental data fusion. Newly arrived data items are directly inserted into a multidimensional scaling layout of existing data in the moving window when the error of the similarity placement is smaller than a given threshold. Once the threshold is exceeded, the whole layout is recalculated. Interactive exploration and user control are not addressed in [24]. Eventriver [15] processes incoming documents of text streams on the fly using a dynamic keyword processing procedure and an incremental text stream clustering algorithm. Temporal clusters of documents are visually depicted as bubbles floating in a time river and individual documents are not visible from the overview of the stream. This is different from our approach where individual documents can be examined within the global temporal and similarity context. Hetzler et al. [11] visualize text collections in a 2D projection space with fresh and stale documents visually distinguished. They apply IN-SPIRE [23] to a dynamic document flow. When new documents are added, the existing vocabulary content is adjusted and the visual result is regenerated. However, the method does not show the animated transition of the view when new documents enter the system. In comparison, our system reveals the evolvement of the stream in fine details with controllable transient animations.

Other than text data, a few visualization systems have been introduced to visually show the evolution of streams in different domains, for instance, in graphs [6], in networks [3], and in stream videos [8][25].

Our algorithm employs Force-Directed Placement (FDP) for visualizing dynamic documents. FDP [7] has $O(N^3)$ complexity which urges researchers to improve its efficiency in order to achieve better computational performance. Intuitively, those improvements come at a price, restrictions are imposed on the force calculations to a subset of the entire data, which could possibly lead to misleading approximated results [4, 17]. Unlike these methods working on static high-dimensional data, our approach is among the first efforts to visualize live text streams using force-directed placement. Furthermore, to make correct dynamic behavior, we avoid applying direct approximation with the reduced force computation scope on only a portion of the particles. Instead, we innovate using a spatial division of the visualization domain (i.e., the similarity grid) for fast locating the appropriate initial position of particles before performing the force computing. This arrangement reduces the simulation time and maintains the accuracy of visualization results. More importantly, we fully utilize the parallel nature of the simulation algorithm by GPU acceleration which achieves a dramatic speedup.

3 System Overview

The infrastructure of STREAMIT is illustrated in Figure 1. Incoming documents of a text stream are continuously fed into a force-based dynamic system, where the documents are represented as dynamically-moving particles in a 2D domain. In this paper, we assume that each document has been characterized by a set of key-

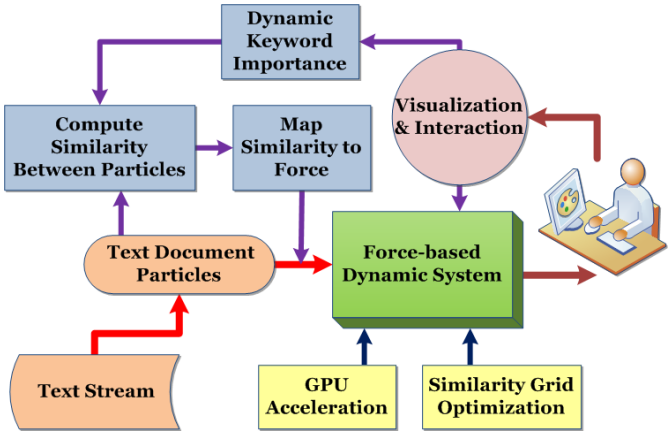

Figure 1: STREAMIT system overview.

words, which are either provided by the text source or automatically generated by approaches that characterize incoming text on-the-fly, such as the technique presented in [16]. The keywords are used to compute pairwise similarity between particles. We then map the similarity into a physical force between each pair of document particles, which drives the motion of particles inside the dynamic system. Upon the arrival of new documents, the particles move until an equilibrium state is reached. The dynamic system is optimized to achieve real time computation for live text streams with moderate scale, through GPU acceleration and utilizing a spatial similarity grid.

The movement of the particles in the dynamic system is visually presented to users. Users can interactively explore the visualization to control browsing, retrieving, and examining documents. Moreover, they can dynamically set keyword importance to influence the computation of similarity and force, and consequently to manipulate the structures and dynamics of the system in particular related to varying user focus and interest.

4 Force-Based Dynamic System

4.1 Particle Potential

Documents are presented as mass particles inside the 2D domain with their velocity and acceleration following Newton's law of motion. Each pair of particles has a potential energy Φ_{ij}:

$$\Phi_{ij} = \alpha(|\mathbf{l}_i - \mathbf{l}_j| - l_{ij})^2, \tag{1}$$

where α is a control constant, and $\mathbf{l}_i$ and $\mathbf{l}_j$ are the positions of two document particles p_i and p_j, respectively. While $|\mathbf{l}_i - \mathbf{l}_j|$ represents the Euclidian distance of the two particles, and l_{ij} is their ideal distance computed from similarity. Hence, this pair potential function models the deviation of the two particles from their ideal locations, which is achieved at zero potential. In numerical computing, the width and height of the 2D domain are set to 1.0.

4.2 Particle Similarity

An optimal layout is determined by the definition of l_{ij}. For text document usually represented as keywords, l_{ij} is obtained from the pairwise similarity computed from their keywords as:

$$l_{ij} = 1 - \delta(p_i, p_j), \tag{2}$$

where $\delta(p_i, p_j) \in [0,1]$ is the cosine similarity between document particles p_i and p_j [9]. With this formula, those documents with large similarity will have a smaller ideal distance, l_{ij}, and move closer for clustering in the visualization.

4.3 Force-Directed Model

For the whole system, it is obvious that not all the particles can be located at ideal locations. A global potential function is the sum of the pairwise energy:

$$V(\mathbf{l}_1, ..., \mathbf{l}_N) = \sum_i \sum_{j>i} \Phi_{ij}, \tag{3}$$

where N is the particle number, and $\mathbf{l}_1, ..., \mathbf{l}_N$ represent the current locations of these particles. The potential of the system is minimized to an equilibrium state that provides a global optimized placement of these particles. A numerical simulation is performed to achieve the optimization by minimizing the global potential with a sequence of simulation time steps. At each time step, the minimization leads to forces acting on each particle:

$$F_i = -\nabla_{\mathbf{l}_i} V(\mathbf{l}_1, ..., \mathbf{l}_N), \tag{4}$$

which attracts or repulses particles from each other. From Newton's law:

$$F_i = m_i a_i \tag{5}$$

where m_i is the mass. We compute the particle acceleration as:

$$a_i = \frac{2\sum_j \alpha(|\mathbf{l}_i - \mathbf{l}_j| - l_{ij})}{m_i}, \tag{6}$$

which is used to update the location of the particle, p_i, at each simulation time step. While every particle no longer moves (in numerical computing, the displacement smaller than a threshold ξ), the system is optimized to its best visual layout. The whole process can be considered as a simplified version of the molecular dynamics simulation with a simple quadratic energy function [20].

Algorithm 1 Dynamic Simulation Algorithm

```
Set the maximum displacement D as a large value
while D > ξ do
  for i = 0 to N - 1 do
    for j = i + 1 to N do
      F_i += 2 * α(|l_i − l_j| − l_ij)
    end for
  end for
  for i = 0 to N - 1 do
    a_i = F_i / m_i
    update the position of this particle
    update maximum displacement D of all particles
  end for
end while
```

Algorithm 1 describes the basic computing procedure of the dynamic system, where we assume every particle has the same unit mass. The constant α is set to an appropriate value (usually 0.01), so that the numerical simulation is stable, i.e., all the particles will not totally move out of the 2D domain or be squeezed to the center of this domain.

5 Dynamic Keyword Importance

Keywords are vital words that highly occur in a document which represent a brief summary as a vector that conveys the topic of that document. The similarity $\delta(p_i, p_j)$ is typically computed by predefined formula, e.g. cosine similarity, from the keyword vector of documents p_i and p_j. However, stream text collections usually span a long period of time. For a real world stream data set, one keyword might excessively appear for a period of time and then fade out, while another one might frequently pop up during the entire period of time. While users typically do not have knowledge

about the topics for the incoming documents, they will change their focus of interests along the stream evolving. Consequently, the definition and computation of similarity should instead be a function of time and adjusted by user input.

To address the challenge, we propose Dynamic Keyword Importance in addition to the computation of $\delta(p_i, p_j)$, which interactively enables the users to manipulate the significance of keywords at any time. The original cosine similarity can be improved as:

$$\delta(p_i, p_j) = \frac{\sum_{k=1}^{K}(w_{ik}I_k)(w_{jk}I_k)}{\sqrt{\sum_{k=1}^{K}(w_{ik}I_k)^2 \cdot \sum_{k=1}^{n}(w_{jk}I_k)^2}} \quad (7)$$

where I_k is the importance of keyword k, K is the number of keywords, and w_{ik} is the weight of keyword k in the document p_i. The classic cosine similarity can be considered as a special case where $I_k = 1$. All the K weights form the keyword vector of this document. The length of the current vector is dynamically updated, so that our system can handle data streams not prerecorded. The weight of keywords is calculated as:

$$w_{ik} = O_{ik} * \log_2 \frac{N}{n_k} \quad (8)$$

where O_{ik} is the occurrence of keyword k inside the document i, N is the total number of documents, n_k is the number of documents that contain the keyword k inside N. The inverse document frequency factor $\frac{N}{n_k}$ favors keywords concentrated in a few documents of a collection, in comparison to the similar high frequency keywords, which are not concentrated in a few particular documents, but instead are prevalent in the whole collection. Please refer to [21] for details of the keyword weight computation.

Users can freely modify the keyword importance through the visual interface, where frequent keywords are presented in an ordered list. Furthermore, the importance can also be determined automatically by the system as follows:

$$I_k = a * O_k + b * (te_k - ts_k) + c * n_k. \quad (9)$$

Here, O_k is the occurrence of keyword k in the current existing documents. te_k is the last time it appears, and ts_k is the first time it appears. $(te_k - ts_k)$ makes the importance larger for aged keyword. n_k makes the importance larger for the keywords appearing in a large number of different documents. Here, a, b, and c are positive constants satisfying $a + b + c = 1$. They are selected to determine how the three factors are preferred. In our experiments, we use $a = 0.3$, $b = 0.3$, and $c = 0.4$. Users indeed can define their preferred keyword importance in a variety of functions for different purposes.

6 Visualization and Interaction

6.1 Visualization

STREAMIT has a main window, an animation control panel, a keyword table, and a set of document tables (see Figure 2):

Main Window: The main window (top left of Figure 2) visually presents the movement of the particles in the 2D domain of the dynamic system through an animated 2D display. Each document particle is represented by a circle. The positions of the circles are decided by the force-directed model so that the similarities among the documents are reflected by the closeness of the positions. As the simulation goes on, the circle positions dynamically changes to reveal the temporal evolution of the stream. A grey scale is used to indicate the age of the documents, namely the older a document is, the darker is its color (see Figure 2 for an example). The sizes of the circles can be mapped to an attribute of the documents. A minimum distance between two particles can be defined in the force-directed model to reduce/eliminate overlaps among the particles.

Animation Control Panel: STREAMIT buffers recent documents falling into a moving time window (named the buffer window) that is larger than the moving window of currently displayed documents. Users can playback the animation within the buffer window to examine the temporal and semantic evolution of the buffered stream in detail. An animation control panel is used to control the playback (see Figure 2(3)). It is similar to the control panels of most movie players and has play and stop buttons and a progress bar. The progress bar maps to the buffer window and the users can move the slider to start the animation from any moment within the buffer window. During any time of the animation, the users can pause the display to examine a moment of the stream in more details or change parameters such as keyword importance. The animation can be resumed after the parameters are changed.

Keyword Table: STREAMIT provides keyword information in a keyword table (see Figure 2(1)). It lists all the keywords characterizing the documents currently displayed, their frequencies in the displayed documents, importance, and colors. The keyword table is updated whenever a new document arrives. Users can interactively sort this table to find frequent keywords and important keywords. They can also interactively change the keyword importance or colors, which will be further discussed in Section 6.3.

Document Tables: Users can click a tab to show one of four document tables (see Figure 2(2)). They display the titles, authors, and timestamps of the following documents respectively: (1) all buffered documents; (2) all documents that are displayed in the main window; (3) documents selected by users; and (4) document groups created by users. The users can sort the documents by their authors or timestamps. They can also click a title to reach the full text of a document.

6.2 Labeling

Labels revealing semantic contents of a collection are desired in text visualization systems. In text streams such as news collections, titles of the documents contain rich semantic information in a condensed manner and thus STREAMIT uses titles as labels of the documents. Since the 2D layout drives similar documents together, severe clutter can be generated if titles of all documents are displayed. We develop a novel labeling algorithm to provide the most recent semantic information to users with user-controllable clutter levels. In particular, documents are divided into groups according to a dissimilarity threshold. Within each group, the dissimilarities among the documents are less than the threshold. Only one document, namely the most recently arrived document, is labeled in each group. A newly arrived document can either be assigned to an existing group or form a new group if it is dissimilar to any existing documents. By interactively changing the dissimilarity threshold and thus changing the grouping, the users can control the clutter caused by labels. In addition, the above algorithm has the benefit that after a document is inserted into the display, existing labels won't be changed except that one label may be removed (the label of the representative document of the group to which the new document is assigned). This is an important feature since we want to keep the consistency among the adjacent displays. The newest injected document will always be labeled according to the algorithm, which is usually desired in text stream visualization. Figure 2 shows the automatic labeling results. In this figure, the newest injected document and its label are highlighted by red (see Figure2(4)) while the selected documents and their labels are highlighted by orange (see Figure2(5)).

Labels and particles may overlap when a large number of documents are displayed. STREAMIT displays labels on the top of particles and allows users to interactively change the transparency of their background. An opaque background makes the labels easy to read and semi-transparent background allows users to examine particles hidden by the labels. In Figure 2, a semi-transparent back-

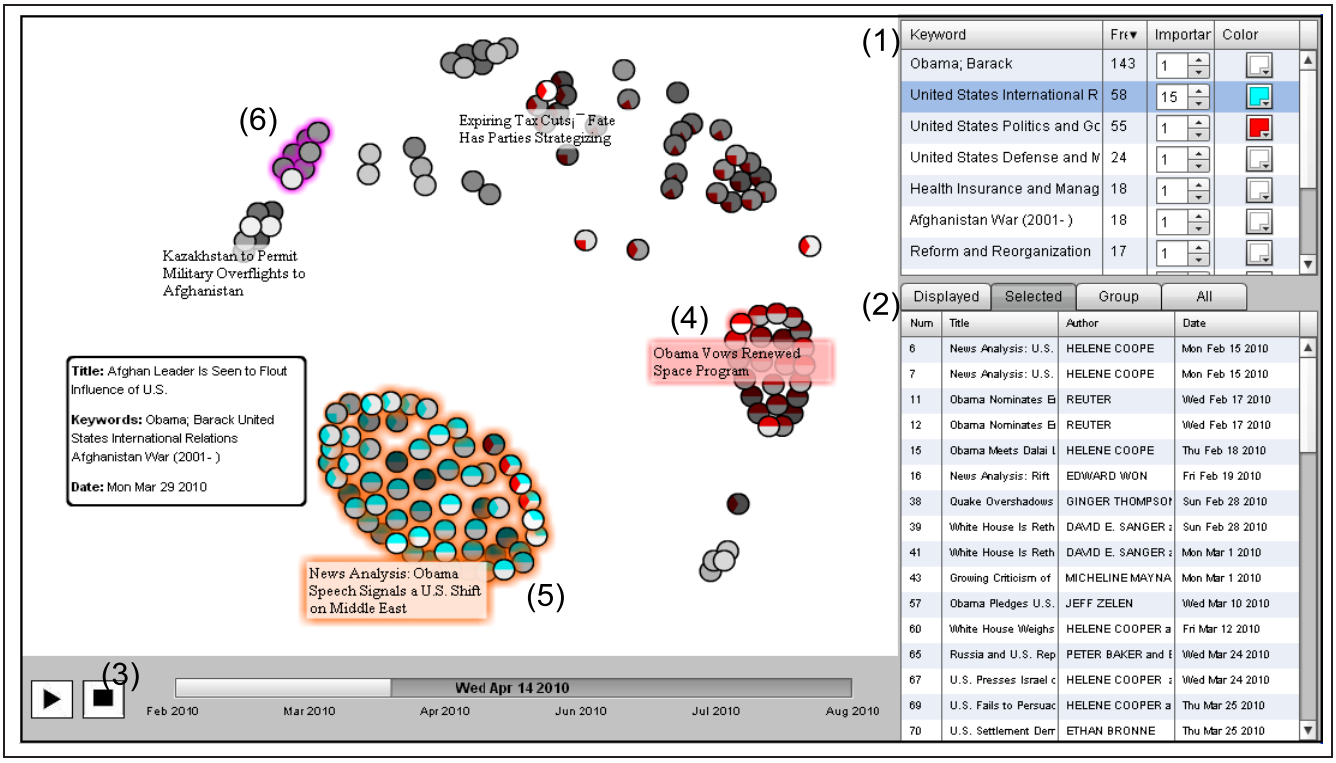

Figure 2: STREAMIT interface. The left part is the visualization view of text streams, and the right part includes keyword table, document tables and parameter controls.

ground is used. Users can turn off all the labels to focus on the colors and layout of the particles. They can also turn on/off the label of an individual particle by clicking it.

6.3 Interaction

STREAMIT allows users to interactively change the display according to their interest. It also allows users to interactively search, track, and examine documents.

6.3.1 2D Display Manipulation

Adjusting Keyword Importance: Users can adjust the keyword importance to emphasize particular topics of current interest. The visualization will immediately respond by representing the movement of the particles after the changes (see Figure 3 for an example).

Grouping and Tracking Documents: When users find a group of interesting documents, they can form a group for them. Documents within a group are highlighted by halos of a color assigned to the group by the users. The halos help to track topics of interest in the animations. Multiple groups can be traced at the same time. Figure 2(5) and (6) show two groups highlighted in orange and pink respectively.

Browsing and Tracking Keywords: Users can assign colors to keywords of interest to track them in the text stream. A document with any traced keywords, no matter if it is an existing document or incoming document, is represented by circular pies where each pie conveys the color of a traced keyword. The size of a pie is proportional to the weight of the keyword in the document. Users can investigate keyword and document relations and track the evolution of relevant topics in this way (see Figure 3 for an example). When the sizes of the pies are small, this approach loses its effectiveness and an alternative approach can be used. In the alternative approach, the users click a keyword of interest in the keyword table. All documents containing the clicked keywords are highlighted by halos so that the users can examine the distribution of these documents in the whole stream segment displayed. The users can sweep the keyword table in this way to find keywords of interest. This approach works well even when the particles are small.

Setting Moving Windows: Users can interactively change the length of the moving window, i.e., investigating period, of currently displayed documents.

6.3.2 Document Selection

Manual Selection: Users can manually select documents from the document tables. They can also use mouse dragging or rubber band to select documents in the 2D display. The selected documents will be highlighted in the main window by halos around the document particles, as shown in Figure 2. Their information will also be displayed in the selected document table (see Figure 2(2)).

Example-based Selection: Users can use the current selection as examples and select documents that are within a distance range to them. The threshold is easily controlled through the keyboard. In this way, users can select a group of similar documents.

Keyword-based Selection: Users can select multiple keywords from the keyword table (see Figure 2(1)), and then the documents that contain the keywords are automatically selected and highlighted (see Figure 2(5)).

Shoebox: In the highly dynamic environment of STREAMIT, users may want to focus on the temporal evolution and examine the selected documents later. They can do it easily by sending the selected documents into a shoebox. The current selection will be cleared so that users can start a new circle of exploration. Later on when the users have time, they can open the shoebox to examine the saved documents in details. Clicking a document in the shoebox will open a new window that contains the full-text of the document.

7 Case Studies

We present two case studies in this section to illustrate how STREAMIT can be used. In the case studies, documents in pre-

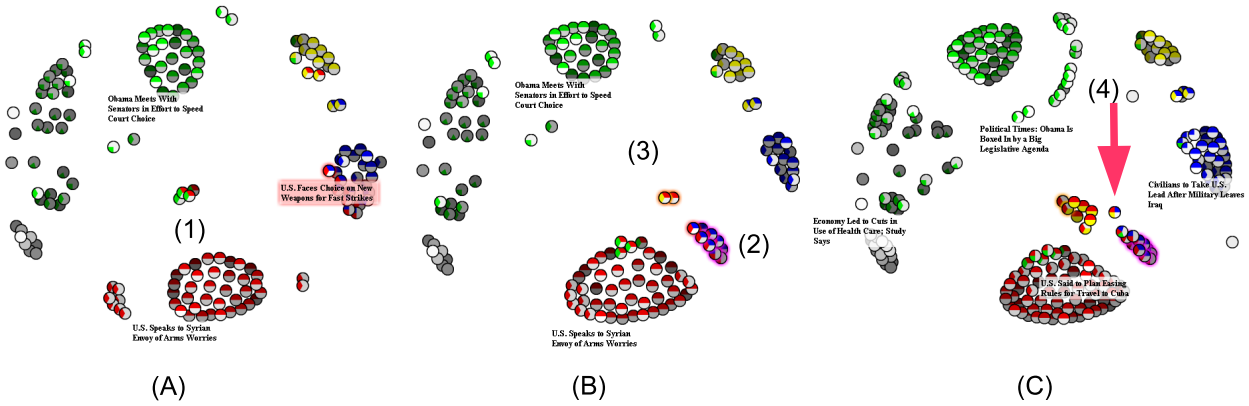

Figure 3: Barack Obama news. (A) Aug. 13, 2010, 136 news articles; (B) after increasing importance of "International Relations"; (C) Sep. 18, 2010, 230 news articles. Keyword colors: "Politics" - green, "International Relations" - red, "Terrorism" - yellow, and "Defense and Military" - blue.

recorded text collections are sorted by their time stamps and fed into STREMIT one by one with an interval of a few seconds to simulate a fast evolving live text stream.

7.1 Case Study 1: Barack Obama News

We use STREAMIT to explore a small text stream simulated using 230 New York Times news (www.nytimes.com) about Barack Obama reported between Jul. 19 and Sep. 18, 2010. The keywords used to characterize the news are tags that come with the news. In each document, the occurrences of the characterizing keywords are assigned to a value of one. The moving window covers the whole stream. Keyword importance is automatically assigned by the algorithm described in Equation 9. As the articles are continuously injected, new keywords are added to the keyword table and their frequencies in the moving window are updated on-the-fly.

Figure 3(A) shows the display on Aug.13, 2010, where 136 news articles are represented. On Aug.13, 2010 (time in the stream), we notice that keywords such as "Politics and Government", "International Relations", "Defences Military", and "Terrorism" have high frequency values according to the keyword table. We consider them as hot topics in the stream and assign them distinct colors to track the news articles characterized by them in the display, as shown in Figure 3(A)..

To emphasize our interest in news about "International Relations" (they are in red), we manually increase the importance of the keyword "International Relations". The resulting display is shown in Figure 3(B). Now news articles containing the keyword "International Relations" are attracted closer to each other than in Figure 3(A). We easily select them using a rubber band selection and find that they contain sub-topics such as "China", "Terrorism" and "Afghanistan War" from the shoebox.

Among the above sub-topics, we want to focus on the topic of "Afghanistan War" and "Terrorism" since most of these news articles recently happened (with lighter darkness). To track the subtopic of "Afghanistan War", we click the keyword "Afghanistan War" to select the related articles and create a new group named "war" for them. We also highlight the group in pink halos (see Figure 3(B-2)). We create another group for the topic "Terrorism" in the same way and highlight them in orange halos (see Figure 3(B-3)). Then we continue to play the animation and track the evolution of these groups. Figure 3(C) shows the visualization when all the news articles are displayed. In the visualization, we notice that the cluster shown in Figure 3(C-3) gets much bigger. We also notice that there is a recent news article (Figure 3(C-4)) that stands in-between it and the cluster shown in Figure 3(C-2). It is related to both "Afghanistan War" and "Terrorism" (see Figure 3(C-4)). We select this article and read it in full detail by clicking the circle.

7.2 Case Study 2: NSF Award Abstracts

In this case, we explore a stream simulated using 1000 National Science Foundation (NSF) IIS award abstracts that were funded between Mar. 2000 and Aug. 2003. Each document was automatically characterized by a set of keywords. The size of a document circle in the display is mapped to the funding amount of the project.

Figure 4 shows several snapshots of the animated visualization. Figures 4(A) and (B) show the stream in two adjacent days. We notice that many large projects (in funding amounts) started from the second day. We pause the animation and select and examine them in detail. From the shoebox, we observe that the keywords "Management" and "Database" appear in many of these project abstracts. It evokes our interest. Therefore, we highlight the keyword "Management" in red and the keyword "Database" in green. We also increase their importance values so that we can observe the relevant abstracts easier (see Figure 4(B)). It can be seen that although some abstracts contain both keywords (see Figure 4(B-1)), there are many other abstracts that contain only one of them. We pull back the animation to the previous day in the same setting (see Figure 4(A)) to examine the temporal evolution of these topics. When the stream further evolves, we observe that IIS continuously supported projects with these keywords (see Figure 4(C)).

Figure 4(C) also illustrates how to discover transformative proposals in a temporal context using STREAMIT. We highlight all projects containing the keyword "sensor" by halos. The node with a halo indicated by the arrow (see Figure 4(C-2)) is a potential transformative proposal since it is far away from the other projects with halos. We select and examine this abstract in detail and learn that it is a project about just-in-time information retrieval on wearable computers.

8 Performance Optimization

Algorithm 1 is a $O(N^2)$ approach. We do not introduce an approximated processing that applies the force computation only in a portion of particles (e.g., in a neighborhood), in order to avoid introducing possible errors. Instead, we seek to optimize the performance of our system by applying parallel computing and similarity grid to improve its scalability for large data sets.

8.1 GPU Acceleration

Our computational algorithm is inherently parallel at each simulation step. Hence, we pleasantly accelerate the computation on graphics hardware with CUDA implementation. N particles execute their kernel program (i.e. the force-placement algorithm) simultaneously as individual threads distributed to a grid of CUDA blocks. Each thread accesses and updates the particle's position from the information of last step which is loaded into the shared memory of

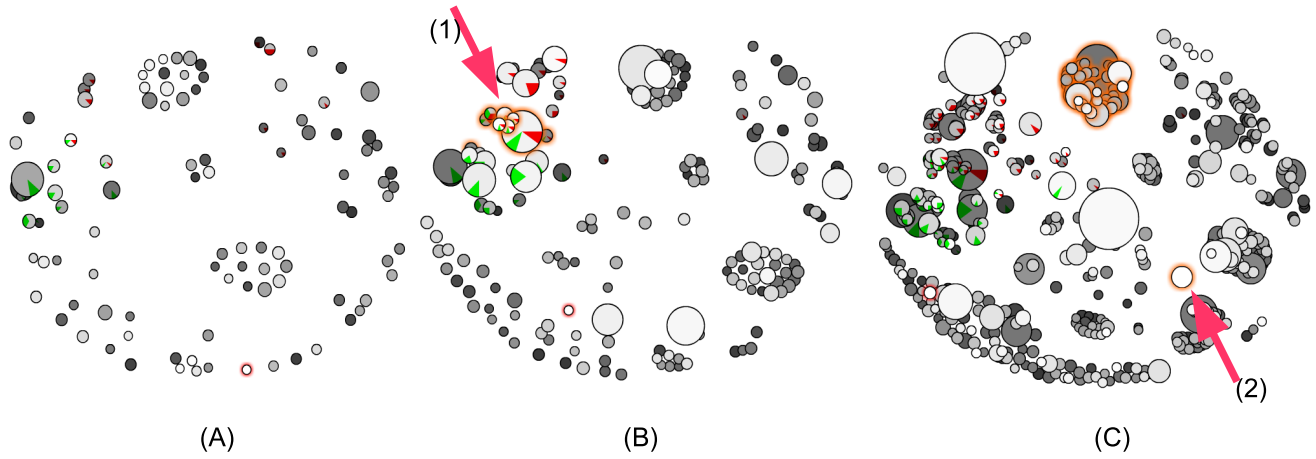

Figure 4: NSF research collections. (A) Aug. 1, 2000, 95 research projects; (B) Sep. 1, 2000, 172 research projects; (C) Mar. 15, 2002, 672 research projects. Keyword colors: "Management" - green, "Database" - red.

Table 1: STREMIT performance on CPU and GPU (in milliseconds) with selected text streams of New York Times news

Document Time Period	Number of Documents in System	Number of Keywords in System	Ave. Simulation Time Per Frame		GPU/CPU Speedup	Maximum Simulation Time		Avg. of Simulation Steps Per Frame
			CPU (ms)	GPU (ms)		CPU (ms)	GPU (ms)	
Feb.13 - Aug.18, 2010	6157	5057	540	34	17.9	4350	230	173
Aug.1 - Oct.31, 2006	7100	1059	620	41	15.1	9070	480	177
Jul.1 - Aug.31, 2010	10205	2036	986	53	15.9	11030	610	200
Synthetic Data set	15000	2000	1020	65	15.7	13070	682	196

the blocks. The solution is similar to an N-body problem which was accelerated on GPU [18].

With the parallel acceleration, we achieved very good performance on consumer graphics cards and PCs for large scale data sets. Table 1 shows the performance evaluated on an NVidia Quadro NVS 295 GPU with 2GB texture memory. In comparison, we also ran the programs on an Intel Core2 1.8GHz CPU with 2GB RAM. We conducted a few experiments using text streams containing news documents from the New York Times news. Incoming documents were inserted into STREAMIT in sequence according to their generation time. Each time when a document was added, the system performed simulation and updated the visualization result accordingly, which formed a frame of the dynamic animation for exploring the text stream. For each frame, the simulation ran multiple steps with the preset minimum threshold ξ, which was set to a very small value, 10^{-4}. We report the number of documents for the experiments, the average simulation time per frame on the CPU and GPU respectively, and the speedup factor. Meanwhile, the maximum simulation time in all frames is reported, which was the maximum waiting time for one visualization update. For each frame, we also recorded the number of computational steps of the simulation. On average, real time running performance was achieved by the GPU acceleration where our system simulated and visualized the text stream at a frame rate around 25-30 frames per second. It was above 15 times faster than the CPU version. Furthermore, the maximum simulation time after document insertion on the GPU was less than a second, which was sufficiently fast considering the relatively slower response time for human perception and analysis of the visualization update. In addition, we tested STREAMIT on a synthetic data set with around 15,000 documents and 2,000 keywords. The results showed that our system worked well for such a large text stream on the GPU.

8.2 Similarity Grid

The initial positions of document particles significantly affect the computational steps and cost of the simulation system. When new particles are inserted into the system, randomly assigning their positions may take many steps for them to move to optimal locations. To address this issue, we employ a similarity grid to ensure that new documents are roughly inserted within the proximity of similar documents. The grid spatially divides the 2D visualization domain into rectangular cells with a given resolution. Each cell has a special keyword vector consisting of the average keyword weights computed from the documents inside the cell. For a new document, we first compute its similarity with this special keyword vector of the grid cells to find the most similar one, and then place this document at the center of that cell. When the system starts, all the cells are empty. The first document is randomly placed, and afterwards, the similarity grid is actively updated. The appropriate resolution of the grid will provide a good acceleration while it cannot be too large due to the extra overhead from the grid maintenance.

Table 2: Performance optimization obtained by employing similarity grids on a data set of 7100 documents

Similarity Grid Size	Average Number of Simulation Steps
None	225
20×20	207
50×50	177
100×100	182
200×200	186

Table 2 shows the average number of simulation steps required on a data set of 7100 documents with different similarity grid sizes. A 50×50 grid decreased the simulation steps per frame to 78% of the steps needed if not using the grid. Meanwhile, the execution time was reduced with the same ratio. This grid resolution is a good choice reducing the simulation steps while keeping the management overhead relatively minimal. Based on the analysis, we used a 50×50 grid for our experiments reported in Table 1.

8.3 Discussion

The performance optimization promotes our system and makes it applicable in a monitoring setting for live streams. In the New

York Times case, news items are produced and arrive in our system continuously, with an averaging 3 documents per hour and a maximum 8 documents per hour at the peak time. The minimum interval between consecutive arrival is around 1 minutes. A capable realtime visualization system should be able to handle newly inserted items faster than this minimum interval. From Table 1, the maximum simulation time of the CPU computing is a few seconds. With GPU acceleration, the handling time is further reduced to less than one second. Therefore, our system can be effectively employed for this live news stream with many thousands of documents accommodated in the display for analysis. It has the ability to directly handle live text streams with document arrival interval around 1 second. Note that the capability is provided with ordinary consumer PC and graphic card.

To further increase the scalability, our system is being actively upgraded, in order to handle real-world text streams with a greater amount of documents and with a smaller arrival interval. We plan to improve the simulation speed by (1) adopting hardware with more computational power, e.g. advanced GPU cards or cluster; (2) exploring hierarchical or multiple-resolution simulation. Meanwhile, we will utilize the unbalanced text streaming speed to provide a management module, which buffers document items during peak times and handle them in idling periods. A very large number of documents inside the system will undoubtedly introduce visual clutters and hinder the ingestion of analyzers. We will also study efficient methods to alleviate cluttering and provide useful abstraction and simplification in visualization.

9 Conclusion

We have presented a new visual exploration system, STREAMIT, for text streams and applied it to visualize collections of news documents. The system employs a physical framework based on interparticle potentials to cluster and analyze incoming document in a dynamic nature. Morerover, we have introduced a new *Dynamic Keywords Importance* that helps users interactively manipulate the importance of keywords for different visualization results. Furthermore, the system is equipped with powerful interactive tools and accelerated on consumer GPUs, consequently providing fast simulation, immediate response and convenient control, which lend itself a good exploratory tool for text stream analysis.

Our STREAMIT system is subject for further research and extension. For instance, we plan to further investigate the mechanism of utilizing keyword importance. We will also develop visualization and simulation schemes suitable for even larger data sets. Moreover, we will extend the system for concurrent visualization with multiple users. In addition, we plan to apply STREAMIT to a variety of real life applications, such as to help research funding managers to explore their document collections and to allow people to explore large news collections through touchable mobile devices.

Acknowledgment

This work is in part supported by National Science Foundation under grant number IIS-0915528, IIS-0916131 and NSF-DACS10P1309. We thank the anonymous reviewers for helpful reviews to improve the paper.

References

[1] C. Albrecht-Buehler, B. Watson, and D. Shamma. Visualizing live text streams using motion and temporal pooling. *IEEE Computer Graphics and Applications*, 25(3):52–59, June 2005.

[2] K. Andrews, W. Kienreich, V. Sabol, J. Becker, G. Droschl, F. Kappe, M. Granitzer, P. Auer, and K. Tochtermann. The infosky visual explorer: Exploiting hierarchical structure and document similarities. *Information Visualization*, 1(3):166–181, Dec. 2002.

[3] U. Brandes and S. Corman. Visual unrolling of network evolution and the analysis of dynamic discourse. *Information Visualization*, 2(1):40–50, 2003.

[4] M. Chalmers. A linear iteration time layout algorithm for visualising high-dimensional data. In *Proceedings of the 7th conference on Visualization '96*, 1996.

[5] Y. Chen, L. Wang, M. Dong, and J. Hua. Exemplar-based visualization of large document corpus (infovis2009-1115). *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1161–1168, 2009.

[6] Y. Frishman and A. Tal. Online dynamic graph drawing. *IEEE Transactions on Visualization and Computer Graphics*, 14(4):727–740, Aug. 2007.

[7] T. Fruchterman and E. Reingold. Graph drawing by force-directed placement. *Software - Practice and Experience*, 21(11):1129–1164, Nov. 1991.

[8] M. Ghoniem, D. Luo, J. Yang, and W. Ribarsky. Newslab: Exploratory broadcast news video analysis. In *Proceedings of the 2007 IEEE Symposium on Visual Analytics Science and Technology*, pages 123–130, 2007.

[9] J. Han and M. Kamber. *Data mining: concepts and techniques, 2nd Edition*. Morgan Kaufmann, San Francisco, CA, USA, 2006.

[10] S. Havre, P. Whitney, and L. Nowell. Themeriver: Visualizing thematic changes in large document collections. *IEEE Transactions on Visualization and Computer Graphics*, 8:9–20, 2002.

[11] E. G. Hetzler, V. L. Crow, D. A. Payne, and A. E. Turner. Turning the bucket of text into a pipe. In *Proceedings of IEEE Symposium on Information Visualization*, page 12, Washington, DC, USA, 2005. IEEE Computer Society.

[12] Y. Ishikawa and M. Hasegawa. T-scroll: Visualizing trends in a time-series of documents for interactive user exploration. *Lecture Notes in Computer Science*, 4675:235–246, Nov. 2007.

[13] J. Leskovec, L. Backstrom, and J. Kleinberg. Meme-tracking and the dynamics of the news cycle. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 497–506, 2009.

[14] S. Liu, M. X. Zhou, S. Pan, W. Qian, W. Cai, and X. Lian. Interactive, topic-based visual text summarization and analysis. In *Proceeding of the 18th ACM conference on Information and knowledge management*, pages 543–552, 2009.

[15] D. Luo, J. Yang, M. Krstajic, J. Fan, W. Ribarsky, and D. Keim. Eventriver: An event-based visual analytics approach to exploring large text collections with a temporal focus. In *IEEE Transactions on Visualization and Computer Graphics*, To appear.

[16] H. Luo, J. Fan, Y. Gao, W. Ribarsky, and S. Satoh. Large-scale news video retrieval via visualization. In *in ACM Multimedia*, pages 783–784, 2006.

[17] A. Morrison, G. Ross, and M. Chalmers. A hybrid layout algorithm for sub-quadratic multidimensional scaling. In *Proc. IEEE Symposium on Information Visualization*, pages 152–158, 2002.

[18] L. Nyland, M. Harris, and J. Prins. Fast n-body simulation with cuda. In *GPU Gems 3*.

[19] F. Paulovich and R. Minghim. Hipp: A novel hierarchical point placement strategy and its application to the exploration of document collections. *IEEE Transaction on Visualization and Computer Graphics*, 16(8):1229–1236, Nov. 2008.

[20] D. C. Rapaport. *The Art of Molecular Dynamics Simulation*. Cambridge University Press, New York, NY, USA, 1996.

[21] G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing and Management*, 24(5):513–523, 1988.

[22] F. B. Viegas, S. Golder, and S. Donath. Visualizing email content: portraying relationships from conversational histories. In *Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 543–552, 2006.

[23] J. A. Wise, J. J. Thomas, K. Pennock, D. Lantrip, M. Pottier, A. Schur, and V. Crow. Visualizing the non-visual: spatial analysis and interaction with information for text documents. pages 442–450, 1999.

[24] P. C. Wong, H. Foote, D. Adams, W. Cowley, and J. Thomas. Dynamic visualization of transient data streams. *IEEE Symposium on Information Visualization*, 0:13, 2003.

[25] J. Yang, D. Luo, and Y. Liu. Newdle: Interactive visual exploration of large online news collections. *IEEE Computer Graphics and Applications*, 30:32–41, 2010.