CompactMap: A Mental Map Preserving Visual Interface for Streaming Text Data

Xiaotong Liu
The Ohio State University
Columbus, USA
liuxiaot@cse.ohio-state.edu

Yifan Hu, Stephen North
AT&T Labs Research
Florham Park, USA
{*yifanhu,north*}*@research.att.com*

Han-Wei Shen
The Ohio State University
Columbus, USA
hwshen@cse.ohio-state.edu

***Abstract*—As text streams become increasingly available from social media such as Facebook and Twitter, visual analysis of streaming text data is playing an important role in most business sectors. A fundamental challenge in visualizing a large amount of streaming text data is to preserve the user's mental map to enable tracking dynamic changes in topics, while simultaneously utilizing the display space efficiently. In this paper, we present CompactMap, an online visual interface that packs text clusters efficiently, with stable updates to maintain the user's mental map. It achieves spatiotemporally coherent layouts by dynamically matching clusters across time, and removing cluster overlaps according to spatial proximity and constraints. We developed a visual search engine based on CompactMaps for exploring a large amount of text streams in details on demand. We demonstrate the effectiveness of our approach in a controlled user study compared with a competing method.**

***Keywords*-Dynamic visualization, mental map preservation, streaming text data, visual search engine**

I. Introduction

Less than 10 years old, Twitter has become a ubiquitous micro-blogging service. It has revolutionized and reshaped the spread of information in the cyberworld. Many organizations now extract and analyze useful nuggets of information from Twitter messages (tweets). Marketing experts and other people who study public sentiment are very interested in analyzing social media messages to quickly gauge the impact of marketing campaigns, and to capture feedback about new products. In real-time analysis scenarios, tweets are often modeled as data streams of small packets or messages that contain unstructured or semi-structured texts. A fundamental challenge for tweet analysis is the scale of the data involved. Until June 2012, Twitter has over 517 million users [20], generating over 340 million tweets and handling over 1.6 billion search queries per day. With the large volume of tweets, it is nearly impossible even to manually explore and find critical information, let alone to monitor and analyze the evolution of important topics over time.

To provide mid-level overviews of tweet streams and topic trends, Gansner *et al.* previously proposed TwitterScope [12]. This system clusters tweet streams into subtopics based on semantic analysis, and renders subtopics as countries using a geographic map metaphor called GMap [13]. While TwitterScope seems effective based on informal trials, several issues were found in visualizing a large amount of tweet streams. First, the irregular shapes of GMap clusters make it difficult to compare cluster sizes to decide which topics are more popular. Second, the clusters often do not fill the display space very well, forcing the layout to be scaled down to fit the display space, which makes pictures and labels harder to see. Third, GMap layouts can have an imbalanced aspect ratio, and periodically repacking afresh to overcome this may significantly change the layout, which disrupts the user's mental map. Also, TwitterScope is limited in only monitoring a fixed set of pre-defined keywords. Arbitrary keyword queries would be more useful, allowing users to search for their own topics of interest.

To solve these problems, we propose *CompactMap*, a visual interface that packs tweet clusters effectively while generating stable layouts. CompactMap achieves coherent layout by dynamically matching clusters across time, and removes overlaps using constrained multidimensional scaling. We also describe an online visual search engine based on CompactMaps for text stream querying and message retrieval. It helps users interactively explore filtered text streams in details on demand. We conducted a controlled user study that shows the effectiveness of CompactMap in comparison with GMap for visual search, comparison and tracking. To summarize, the main contributions of this work are:

- **A dynamic visualization technique** that displays clusters in text streams as stable, space-efficient layouts.
- **A real-time visual search engine** that supports arbitrary keyword search combined with semantic analysis of topics.
- **An enhanced real-time visual analysis system** that enables users to explore and compare discussions and topics of interest intuitively and flexibly.

II. Related Work

Social media data monitoring and analysis has been increasingly popular as the influence of social media grows in daily life. Dork *et al.* [10] introduced Visual Backchannel as a way of following and exploring online conversations about large-scale events. Cuvelier and Aufaure [7] proposed topographic networks of tags, representing a tag cloud with a topographic metaphor to highlight the most important

concepts found for a given search on Twitter. Based on off-line processing, Twitinfo [19] employed a timeline-based display to highlight peaks of high tweet activity discovered by a streaming algorithm. Whisper [6] traced and visualized the process of information diffusion in social media using a sunflower metaphor. Previous work by Gansner *et al.* [12] summarized and visualized clustered tweets using a geographical map metaphor [13]. In contrast with the above tools, we aim at providing an online visual search engine that automatically retrieves tweets and summarizes topics for user specified search query, and visualizes the results in a stable view that allow users to track the topics of discussion over time.

For maintaining dynamically stable layouts, most prior work has focused on studying dynamic graph layouts. Brandes and Wagner [5] adapted a force-directed model to dynamic graphs using a Bayesian framework. Diehl and Görg [9] considered graphs in a sequence to create smoother transitions. Brandes and Corman [3] proposed a system for visualizing network evolution, in which mental map preservation was achieved by pre-computing good locations for the nodes and fixing those positions across the network layers. Brandes and Mader [4] recently studied offline dynamic graph drawings and compared various techniques such as aggregation, anchoring and linking. While work on dynamic graph visualization typically assumes the input graph is connected, stable packing of dynamic and disconnected graphs has also been considered [12]. Although graph-based dynamic layouts can achieve relatively stable views, the utilization of display space can be poor due to the arbitrary shape of graph layouts and having too much empty space among nodes. On the other hand, space-filling visualization techniques such as Treemap [2], [16] and Bubblemap [1] can utilize the available space efficiently, but it is often difficult to maintain a dynamic layout over time. Hybrid methods that combine the graph and space-filling layouts were applied to hierarchical data visualization [8], [15] and multiple-category visualization [14]. However, although the underlying rectangle-packing algorithm [15] outperformed Treemap for maintaining the dynamic layout, in the worst case it can still show large displacements of clusters between consecutive frames, as well as considerable wasted space between clusters.

III. Visualization System Design

In this section, we outline a set of challenges for designing an effective visualization system for streaming text data analysis, and give an overview of a prototype system.

A. Design Challenges

First of all, the main challenge is how to handle the high volume and velocity of text streams. Since the real-life data analysis mostly focuses on the data of interest, it will highly reduce the data size for analysis if the system allows analysts to retrieve interesting data on request. To address this, a scalable search engine that can deal with real-time querying and incremental updates across distributed devices is highly desirable.

Secondly, text streams that are close in time often exhibit similar characteristics. Analysis of the content can be helped greatly if the related streams can be appropriately clustered, categorized and summarized. Therefore, fast algorithms are needed to perform semantic analysis and content summarization in real-time.

Thirdly, to allow analysts to monitor and track relevant discussions intuitively, visualizing search results with effective visual encoding schemes is essential to providing an informative, convenient and pleasant data exploring and communicating environment. The main issue to address is how to merge each stream with existing data and update the cluster visualization in a way that preserves the viewer's mental map. Another issue is how to efficiently utilize the display space without scaling down the visualization, which can make pictures and labels hardly legible.

B. System Overview

Figure 1 shows an overview of our visualization system. The server cluster consists of several web servers with different functionality: the streaming server, the storage server, the analysis server and the communication server. By way of Twitter's streaming API, the streaming server collects data with respect to a user-specified keyword query. Unless the user changes the search query, the relevant data is retrieved continuously and stored persistently in the storage server, until the client browser navigates away from visualization. This supports queries over any time period after the streaming server started service. To analyze the sematic contents of the text streams, similarity analysis and tweet clustering are performed on the analysis server, in the same way as TwitterScope [12] did. The content summarization results are then sent to the visualization client (a browser that supports HTML5) via the communication server.

Once the visualization client obtains the newly processed data from the communication server, it visualizes the results using a visual metaphor called *CompactMap*, a dynamic visualization that packs topic clusters within the display space. CompactMap provides a stable view of the time-varying clusters that allows users to visually track the discussion of topics over time, which will be described in the next section.

We integrated CompactMap into the TwitterScope framework to develop an online visualization system for tweet streams (at http://tibesti.research.att.com/twitterscope/). Our visualization system has been online for a few months, and the heaviest tweet rate encountered was less than one million tweets per hour. Our system can efficiently handle concurrent search queries to cope with this packet rate, with implementation of Node.js [17], an event-driven

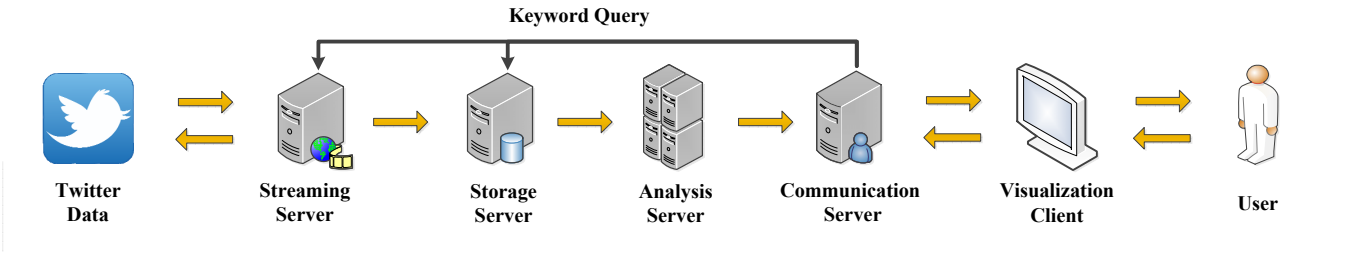

Figure 1. Overview of the visualization system.

environment based on Googles V8 JavaScript engine. The implementation of CompactMap in Javascript and HTML5 is able to update the dynamic visualization of streaming tweet clusters with staged animation in real-time.

IV. CompactMap Visualization

In this section we will describe the visual metaphor and encoding of CompactMap, and present algorithms to generate them.

A. Visual Metaphor and Encoding

A major challenge in visualizing tweet stream clusters is preserving visual stability to help users track dynamic behavior over time. Moreover, this must be done without sacrificing too much display space. Space-filling visualization techniques, such as Treemap [2], [16] and Bubblemap [1] are very good at utilizing the available space, but as far as we know, these techniques have not been extended effectively to cope with dynamic data. Our approach is to devise a dynamic visualization method that integrates a space-filling metaphor with incrementally stable dynamic graph layout.

CompactMaps were inspired by quantum Treemaps [2] and clustered graphs. In CompactMap, clusters are rendered as rectangular nodes. Individual tweets are represented by the Twitter icons that fill the clusters. We chose this metaphor for several reasons. First, regular shapes such as rectangles make it easier to compare cluster sizes, a task which can be difficult using the cartographic map metaphor (Figure 2 (left)) in TwitterScope [12]. Second, the irregular shapes of clusters in a map metaphor often work against effective use of the display space, making pictures and labels less legible after scaling down the visualization to fit the available space. Finally, representing clusters as rectangular nodes allows us to develop spatiotemporally-coherent layout algorithms for dynamic visualization, and at the same time retain the main advantages of Treemaps as a space-filling visualization.

Figure 2 (right) shows an example of a CompactMap visualization. A list of the top words in the tweets of each cluster is drawn as an overlay on each rectangle as a topic summary. Compared with GMap (Figure 2 (left)), the cluster shapes in a CompactMap are more regular, and the icons are larger. A controlled user study comparing these two visualizations will be described in Section V.

B. Dynamic Visualization

In an online social network such as Twitter, topics are dynamic and very diverse. Table I summarizes the basic cases and the possible visualization adjustment. These basic cases can be combined to form more complex behaviors. Note that in our visualization, due to the limit of screen space, we maintain a moving window of up to 400 most recent tweets, thus as new tweets arrive, old tweets are removed and the current set of tweets are reclustered, which give rises to the cases in this table.

Given a stream of clusters, our goal is to preserve the spatiotemporal coherence of the clusters spanning multiple time steps during whatever layout adjustment for dealing with the dynamic topic behavior. Given an initial layout at $t = 1$, our algorithm progressively generates layouts for $t \geq 2$ with spatiotemporal coherence.

The initial layout at $t = 1$ can be any random layout, but we opt to use a spiral-based space filling layout to place the big clusters near the center of the display. We first rank the clustered nodes at $t = 1$ in decreasing order of cluster size, then place the nodes one at a time from the center of the display space along a rectangular spiral of increasing radius. A node's placement is valid if it does not overlap any node already placed; otherwise, it is moved to the next candidate position. Figure 3 (a) shows an initial layout, where the larger clusters are surrounded by smaller ones.

For time steps t and $t+1$, out-going and in-coming clusters have no temporal coherence. Therefore, the visual stability of clusters depends only on how the persistent clusters are adjusted. We consider a cluster to be a *persistent cluster* at t if at least one tweet in that cluster at t still exists in the new stream at $t+1$. From Table I, we know that one cluster can split into multiple ones, and several clusters can merge into a single one. The problem to address is matching the persistent clusters across time to appropriately deal with different dynamic behavior, preserving the user's mental map. We note that the initial spiral layout will not be necessarily maintained after cluster matching, due to the mental map preservation.

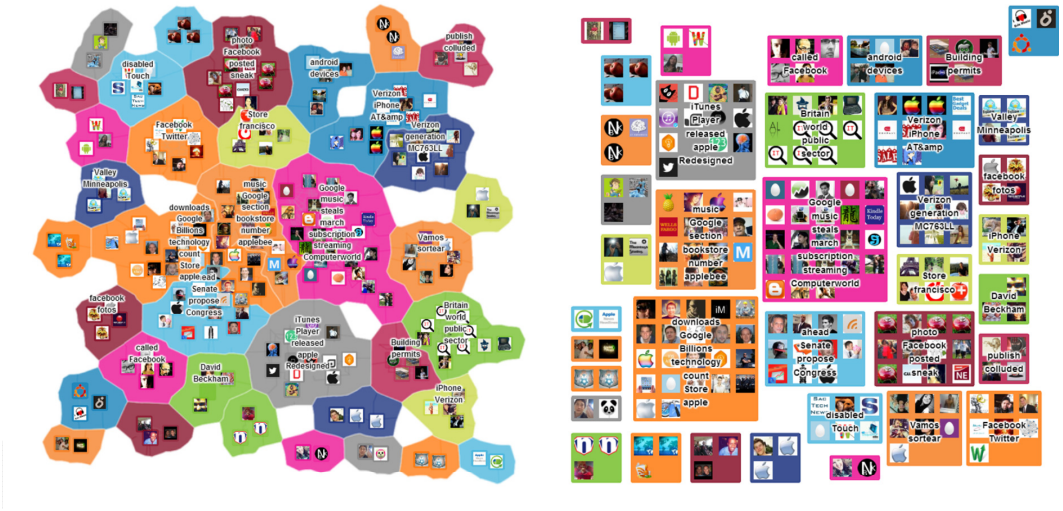

Figure 2. GMap visualization (left) and CompactMap visualization (right) of the tweets for the search term "technology" on May 16, 2013.

Table I
THE BASIC CASES OF DYNAMIC BEHAVIOR OVER TIME AND THE POSSIBLE VISUALIZATION ADJUSTMENT.

Dynamic Topic Behavior Over Time	**Possible Visualization Adjustment**
A topic will not be discussed	Discard its cluster and its tweets
A topic becomes more/less popular	Adjust the size and location of its cluster
A topic splits into multiple subtopics	Separate its cluster into multiple clusters
Several topics merge into one topic	Merge the clusters into one cluster
A new topic emerges	Place the new cluster in an appropriate location

Let g_i^t denote the i-th cluster node at time t, its position p_i^t, and color c_i^t. Let $\phi : g_i^{t+1} \rightarrow \{g_j^t | g_i^{t+1} \cap g_j^t \neq \emptyset\}$ be a one-to-many mapping for the persistent clustered nodes from time step $t+1$ to the previous one t. Our goal is to determine positions and colors of the new clusters such that p_i^{t+1} is as close to those of $\phi(g_i^{t+1})$ as possible. Likewise c_i^{t+1} are the same to at least one the member of $\phi(g_i^{t+1})$, so as to maintain visual stability during the transition.

We first measure the similarity of two cluster nodes (g_i^{t+1}, g_j^t) as:

$$s_{ij} = \left| g_i^{t+1} \cap g_j^t \right|. \tag{1}$$

Two clusters are considered very similar if they share many common elements. With this metric, the position p_i^{t+1} is matched to a weighted centroid of the nodes from the previous time step:

$$p_i^{t+1} = \frac{\sum_{j:g_j^t \in \phi(g_i^{t+1})} s_{ij} p_j^t}{\sum_{j:g_j^t \in \phi(g_i^{t+1})} s_{ij}}. \tag{2}$$

Specifically, when several clusters merge into a single one, the position of the new cluster will be closer to the ones that have more tweets in common at the previous time step; when one cluster splits into multiple ones, the positions of the new clusters are set to the position of the previous cluster. Overlap removal after cluster matching will be discussed in the next subsection.

For the color matching, the color c_i^{t+1} is matched to the most similar one c_j^t from the previous time step:

$$c_i^{t+1} = c_j^t, \text{ where } j = \arg \max_{k:g_k^t \in \phi(g_i^{t+1})} s_{ik}. \tag{3}$$

C. Dynamic Layout

After matching the positions and colors of the clusters that persist from time step t to $t+1$, the next problem is to adjust the layout of clusters to avoid overlap, while maintaining visual stability and efficient space utilization.

We first model the spatial proximity of the matched clusters as a proximity graph $G = (V, E)$ using Delaunay triangulation, with V the set of nodes and E the set of edges. Given a display region Γ, the goal is to find coordinates p_i for each node $i \in V$, such that: (1) there is no overlap between any nodes $\{i, j\} \in V$; (2) each edge $(i, j) \in E$ is close to its ideal length; and (3) each p_i is inside Γ. In CompactMap, nodes are not just points but clusters that have finite area, so the actual display area Γ in relation to a cluster must be shrunk by a margin that is half of that cluster's width and height, so that every cluster can be fully displayed if

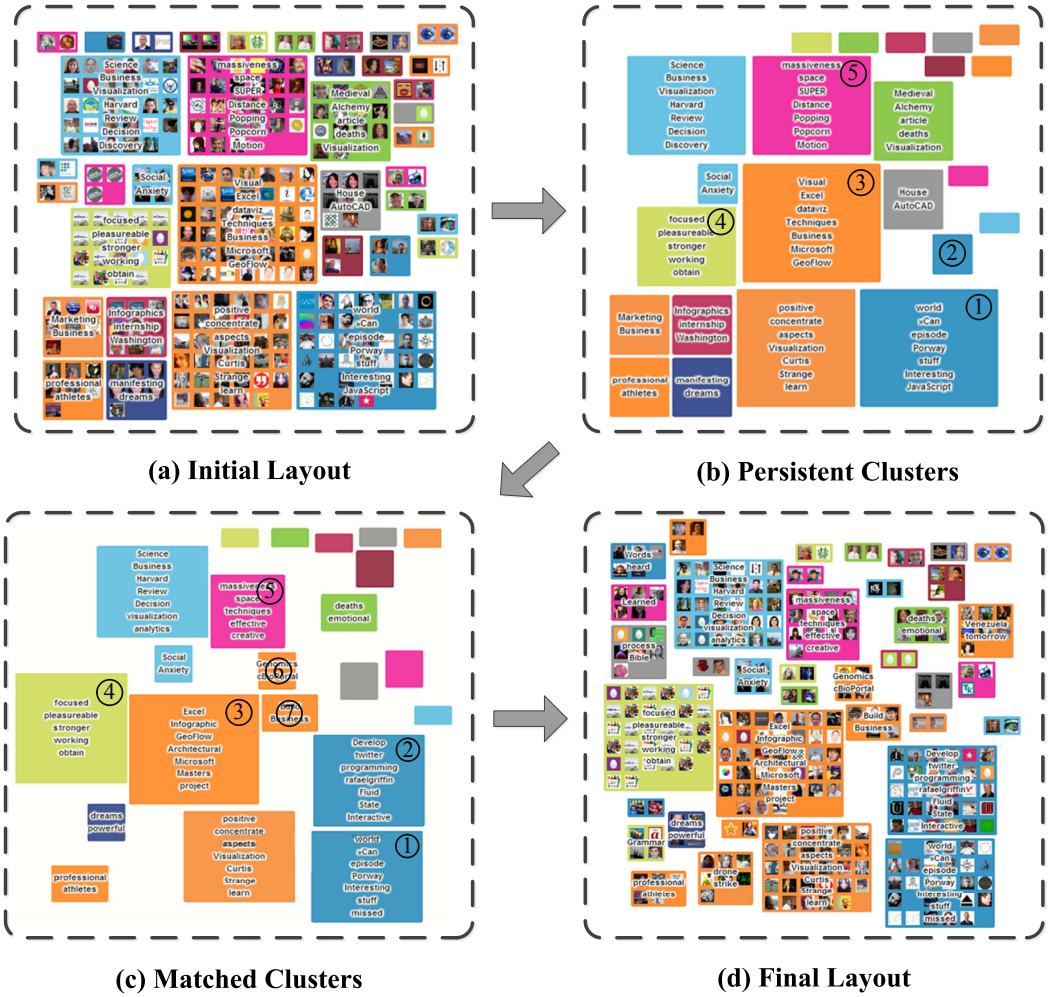

Figure 3. Computation of the dynamic layout for CompactMap: (a) embedding an initial CompactMap in a spiral-based space filling layout; (b) removing outdated clusters and retaining the clusters across time; (c) matching the persistent clusters from the previous time step to the new one with spatial and color coherence, and adjusting the layout by the CMDS method for visual stability and efficient space utilization; (d) incorporating the new clusters to form a new CompactMap.

its center lies inside Γ. With these conditions in mind, we solve the layout problem using constrained multidimensional scaling (CMDS)[1]:

$$\min \sum_{(i,j)\in E} w_{ij}(\|p_i - p_j\| - d_{ij})^2 + \alpha \sum_{p_i \notin \Gamma} (\|p_i - \Gamma(p_i)\|)^2, \quad (4)$$

where E is the set of triangulation edges, d_{ij} is the ideal distance between nodes i and j, w_{ij} is a weighting factor (typically $1/d_{ij}^2$), $\alpha \geq 0$ is a balancing factor, and $\Gamma(p_i)$ denotes the projection of p_i to the region Γ – if p_i is outside of Γ, the projection $\Gamma(p_i)$ gives the point on the boundary of Γ that is closest to p_i; otherwise $\Gamma(p_i) = p_i$. The rigidity of the triangulation provides sufficient scaffolding to constrain the relative positions of the components, and helps preserve the global structure of the original spatial proximity (similar to the PRISM approach [11]). For each edge (i, j), the amount of overlap on the line $x_i \rightarrow x_j$ is denoted by δ_{ij} (δ_{ij} is set to 0 if no collision is detected), and the ideal distance is set as $d_{ij} = l_{ij} + \delta_{ij}$, where l_{ij} is the length of the triangulation edge between node i and j.

We employ the following iterative process to solve (4),

$$p_i \leftarrow \frac{\sum_{(i,j)\in E} w_{ij}(p_j + d_{ij}\frac{p_i - p_j}{\|p_i - p_j\|}) + \alpha\Gamma(p_i)}{\sum_{(i,j)\in E} w_{ij} \; + \; \alpha}, \quad (5)$$

which is easier to implement in languages like JavaScript that do not have sophisticated numerical libraries. Furthermore, by rendering the iterative process, a visually stable animation can be shown, from the initial configuration to a

[1]For a full description and evaluation of the CMDS model, see Liu *et al.* [18], where the model is also applied to laying out small multiples on a grid.

final constrained layout.

To present a consistent view for users to track changes of clusters, staged animation is applied to first remove the outgoing clusters towards the bottom of the view, then match and adjust the positions of the persistent clusters, and finally move the incoming new clusters into the view. The default update rate is every 1 minute, but our algorithm can adapt to any time interval, as long as the client can obtain the newly processed data from the server within the specified time interval.

Figure 3 illustrates the dynamic visualization and layout of CompactMap from time step t to the next one $t+1$. Starting from an initial layout (Figure 3(a)), it first removes outdated clusters. Next, persistent clusters in the previous time step (Figure 3(b)) are matched to those at the next time step (Figure 3(c)), with their layout adjusted by the CMDS method to remove overlaps. From the staged animation, we observed that the previous cluster g_1^t (in Figure 3(b)) split into two parts — one of them was merged with the previous cluster g_2^t to form the new cluster g_2^{t+1} (in Figure 3(c)), and the other stayed as the new cluster g_1^{t+1}; the previous cluster g_3^t split into three clusters (g_3^{t+1}, g_6^{t+1}, g_7^{t+1}) at the next time step; and some clusters such as g_4^t and g_5^t were matched to the new ones g_4^{t+1} and g_5^{t+1} with either a smaller or larger size. Importantly, comparing Figure 3(b) and (c), we can see that the spatial proximity of the matched persistent clusters were well preserved by CMDS, and none of them fell outside the display space, thus providing a visually stable layout during the view transition. Figure 3(d) shows the final layout at the time step $t+1$, where incoming clusters were incorporated into the matched persistent clusters. An animated demonstration of the dynamic visualization and layout of CompactMap is available at http://vimeo.com/67949445, or in the video that accompanies this paper.

V. User Study

We performed a controlled experiment to evaluate the effectiveness of CompactMap visualization for visual search, comparison and tracking, compared with a GMap visualization as described in TwitterScope system [12]. We recruited 15 subjects (10 males, 5 females) having various backgrounds in computer science, geographic information science, electrical engineering, chemical engineering, physics, economics and finance. The subjects ranged in age between 22 and 32 years, with a mean of 25.

A. Tasks and Procedure

We defined three tasks to assess the effectiveness of CompactMap for both static layout (for visual search and comparison) and dynamic layout (for visual tracking):

[T1] Rank three marked clusters by size.

[T2] Count how many times a given icon appears in a marked cluster.

[T3] Determine whether a marked cluster persists at the next time step.

A time limit of 60 seconds is imposed, corresponding to TwitterScope's default update rate.

For both GMap and CompactMap, we created 12 datasets (4 for each task) of tweet clusters with various keyword queries. Each dataset contains 30 to 50 tweet clusters of two continuous time steps. The study was conducted as a within-subjects experiment with 2 experimental conditions (GMap and CompactMap), 3 tasks (**T1**, **T2**, **T3**) and 4 repetitions (visualization image) for each condition. For each repetition, the subject was presented with only one condition. We counter-balanced the selection of conditions in the repetitions so that each subject performed the same number of repetitions for both conditions. The order of the repetitions was random in the study. Figure 2 shows a repetition for **T1** or **T2**, while Figure 4 illustrates a repetition for **T3**.

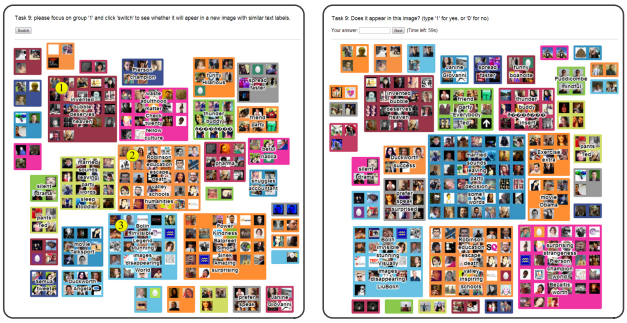

Figure 4. CompactMap visualizations used in the user study system for task **T3** — *determine whether a marked cluster persists at the next time step*. An initial visualization (left) would transition to a new one at the next time step (right) when the subject clicks a "switch" button.

The study was performed on an Intel i7-2600, 3.4 GHz CPU desktop computer and an ordinary 24-inch, 1920 × 1080 pixel display. Prior to the experiment, the subjects viewed a tutorial that provided several training tasks to help them become familiar with the user interface of the experimental system. For each task, the subject was given a randomly chosen GMap or CompactMap visualization, and then prompted to answer the question. After typing in the answer(s) and clicking on the "next' button, the next task was loaded. For task **T3**, subjects were asked to click a "switch" button, and a new image corresponding to the next time step would be displayed. After subjects finished all tasks, they were asked to rate their satisfaction with GMap and CompactMap on a questionnaire containing 5 questions, and finally, to participate in a semi-structured interview. The user study system is presented in a video available at http://vimeo.com/67949445.

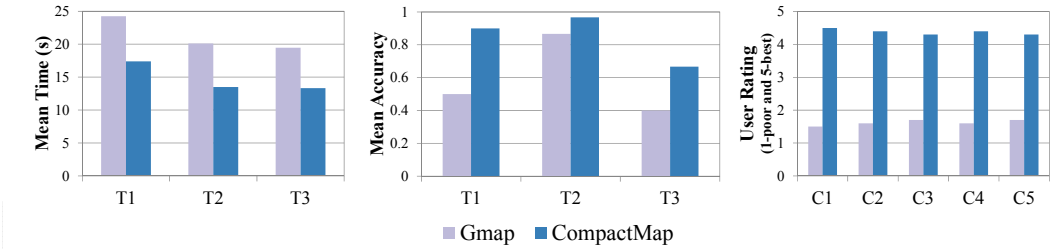

Figure 5. Mean task completion time (left), accuracy (middle) and subjective satisfaction (right) for the user study.

B. Hypotheses

[H1] *CompactMap outperforms GMap in visual comparison of clusters.* Because irregular shapes are difficult to compare by size, we expect CompactMap's rectangular clusters to facilitate visual comparison.

[H2] *CompactMap outperforms GMap in visual search for specific elements.* Because clusters in irregular shapes may also waste display space, the icons in CompactMap will be more readable.

[H3] *CompactMap better supports visual tracking than GMap.* Because the layout of GMap can have an unbalanced aspect ratio, and repacking afresh may significantly change the map layout, CompactMap's dynamic updates better preserve a user's mental map.

[H4] *CompactMap has a positive impact on user satisfaction.* Because rectangular shapes are easier to understand and follow, users will prefer CompactMap to GMap for visualization tasks.

C. Results and Discussion

For each task, we measured accuracy and completion time, and the subjective assessment from the evaluation questionnaires. Task completion time and accuracy measures were evaluated using single factor Analysis of Variance (ANOVA) for the dependent variables. The average completion time and accuracy for each task are shown in Figure 5 (left) and (middle) respectively.

For **T1**, we found a significant effect for task completion time, $F(1,28) = 5.375, p = 0.003$. The average time spent was 24.25 seconds ($SD = 8.36$) with GMap, and 17.40 seconds ($SD = 3.61$) with CompactMap. Our analysis also revealed a significant effect on the accuracy, $F(1,28) = 3.333, p = 0.031$. Overall, the subjects had 50% ($SD = 0.38$) accuracy with GMap and 90% ($SD = 0.21$) with CompactMap.

For **T2**, ANOVA analysis revealed a significant effect for both task completion time ($F(1,28) = 5.233, p = 0.004$) and accuracy ($F(1,28) = 3.143, p = 0.040$). On average, the subjects spent 20.10 seconds ($SD = 4.93$) searching the given target in GMap, while 13.49 seconds ($SD = 2.16$) in CompactMap; 86.67% of the answers were correct with GMap, while 96.67% with CompactMap.

For **T3**, we found a significant effect for task completion time, $F(1,28) = 3.902, p = 0.016$. The average completion time was 19.46 seconds ($SD = 6.34$) with GMap, and 13.33 seconds ($SD = 3.21$) with CompactMap. Task accuracy was not found to have a significant difference, $F(1,28) = 0.471, p = 0.172$. On the whole, the subjects had 40% ($SD = 0.28$) correct answers with GMap, and 66.67% with CompactMap.

The questionnaire asked subjects to assess their satisfaction with CompactMap and GMap on multiple criteria: [**C1**] *GMap/CompactMap was MORE helpful in answering the questions*; [**C2**] *I could find targets MORE quickly with GMap/CompactMap*; [**C3**] *I was MORE confident in my answer with GMap/CompactMap*; [**C4**] *GMap/CompactMap would be MORE beneficial if I used it for visual comparison*; and [**C5**] *GMap/CompactMap is visually MORE pleasing*. The score scale is from 1 (poor) to 5 (best), with 3 as neutral. Figure 5 (right) provides the average ratings for each criterion. Generally CompactMap was rated higher than GMap. In particular, subjects were more efficient and confident in their results with CompactMap.

In interviews, most subjects expressed a preference for the CompactMap design, stating that and it is easier to determine which topics are more popular, compared with GMap. Three subjects mentioned that searching targets in GMap was tedious and even irritating since icons in GMap are not aligned uniformly as in CompactMap. One subject reported that he sometimes counted a single item more than once in GMap, due to the lack of alignment. Several subjects mentioned that GMap initially caused confusion because they interpreted it as a real map at first sight, and wondered if the map showed the geo-locations of tweets.

Based on these results, the hypotheses are well supported. The shorter task completion time and higher accuracy support **H1**, **H2** and **H3**. Statistically, the differences of completion time and accuracy for all tasks were significant, except for the accuracy of **T3**. **H4** is supported by the questionnaire results, which indicates that most subjects preferred the rectangular shapes and dynamic layout provided by CompactMap.

We note that with CompactMap, we lose the ability

to encode cluster-cluster similarity through proximity of clusters, something that GMap can achieve. However, due to the dynamic streaming nature of our data, users typically do not have enough time from one frame to the next to study cluster-cluster relationships, thus, we believe that similarity among clusters is not as important as the benefits CompactMap brings, as discussed in Section IV-A.

In summary, user studies show that CompactMap is useful in helping users in visually searching, comparing and tracking clusters quickly and accurately. The regular shape of clusters was valued by the users, and most users preferred CompactMap.

VI. Conclusion and Future Work

In this study, we presented CompactMap, an online visual interface that packs text clusters in a way that is space-efficient, and generates stable layouts of evolving clusters. It does this by dynamically matching clusters across time, and adjusting the layout according to spatial proximity and constraints that preserve a user's mental map. A controlled user study was conducted to show the effectiveness of CompactMap in comparison with GMap for visual search, comparison and tracking. We implemented CompactMap in a visualization system that provides an online visual search engine for text stream querying and retrieval of items of interest.

In the future, we would like to investigate how to automatically select critical time frames for better understanding of evolving topics. We also hope to make the general keyword search service available to the public as soon as possible.

References

[1] B. B. Bederson. Photomesa: a zoomable image browser using quantum treemaps and bubblemaps. In *Proceedings of the 14th annual ACM symposium on User interface software and technology*, 2001.

[2] B. B. Bederson, B. Shneiderman, and M. Wattenberg. Ordered and quantum treemaps: Making effective use of 2d space to display hierarchies. *ACM Transactions on Graphics*, 2002.

[3] U. Brandes and S. R. Corman. Visual unrolling of network evolution and the analysis of dynamic discourse. *Information Visualization*, 2003.

[4] U. Brandes and M. Mader. A quantitative comparison of stress-minimization approaches for offline dynamic graph drawing. In *Proceedings of the 19th international conference on Graph Drawing*, 2012.

[5] U. Brandes and D. Wagner. A bayesian paradigm for dynamic graph layout. In *Graph Drawing*, 1997.

[6] N. Cao, Y.-R. Lin, X. Sun, D. Lazer, S. Liu, and H. Qu. Whisper: Tracing the spatiotemporal process of information diffusion in real time. *IEEE Transactions on Visualization and Computer Graphics*, 2012.

[7] E. Cuvelier and M.-A. Aufaure. A buzz and e-reputation monitoring tool for twitter based on galois lattices. In *Proceedings of the 19th international conference on Conceptual structures for discovering knowledge*. Springer-Verlag, 2011.

[8] W. Didimo and F. Montecchiani. Fast layout computation of hierarchically clustered networks: Algorithmic advances and experimental analysis. In *The 16th International Conference on Information Visualisation*, 2012.

[9] S. Diehl and C. Görg. Graphs, they are changing. In *the 10th International Symposium on Graph Drawing*, 2002.

[10] M. Dork, D. Gruen, C. Williamson, and S. Carpendale. A visual backchannel for large-scale events. *IEEE Transactions on Visualization and Computer Graphics*, 2010.

[11] E. R. Gansner and Y. Hu. Efficient node overlap removal using a proximity stress model. In *Graph Drawing*. Springer-Verlag, 2009.

[12] E. R. Gansner, Y. Hu, and S. North. Visualizing streaming text data with dynamic graphs and maps. In *Proceedings of the 20th international conference on Graph Drawing*. Springer-Verlag, 2012.

[13] Y. Hu, E. R. Gansner, and S. Kobourov. Visualizing graphs and clusters as maps. *IEEE Computer Graphics and Applications*, 2010.

[14] T. Itoh, C. Muelder, K.-L. Ma, and J. Sese. A hybrid space-filling and force-directed layout method for visualizing multiple-category graphs. In *IEEE Pacific Visualization*, 2009.

[15] T. Itoh, Y. Yamaguchi, Y. Ikehata, and Y. Kajinaga. Hierarchical data visualization using a fast rectangle-packing algorithm. *IEEE Transactions on Visualization and Computer Graphics*, 2004.

[16] B. Johnson and B. Shneiderman. Tree-maps: a space-filling approach to the visualization of hierarchical information structures. In *Proceedings of the 2nd conference on Visualization*, 1991.

[17] Joyent. Node.js, http://nodejs.org/.

[18] X. Liu, Y. Hu, S. North, T.-Y. Lee, and H.-W. Shen. Correlatedmultiples: Spatially coherent small multiples with constrained multidimensional scaling. *OSU Technical Report SERIES (OSU-CISRC-4/13-TR10), ftp://ftp.cse.ohio-state.edu/pub/tech-report/2013/TR10.pdf*, 2013.

[19] A. Marcus, M. S. Bernstein, O. Badar, D. R. Karger, S. Madden, and R. C. Miller. Twitinfo: aggregating and visualizing microblogs for event exploration. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 2011.

[20] Semiocast. Twitter reaches half a billion accounts, with more than 140 millions in the u.s.