TRENDTRACKER: A System for Visualizing Trending Topics on Twitter

Akshay Kannan, Jeff Patzer, Boaz Avital

LIST OF FIGURES

I. INTRODUCTION

Modern web technologies have enabled an abundance of live data streams on the web, such as social network streams, financial market data, and streaming live video. While displaying one dynamic stream in a confined space is relatively easy to do, a major challenge exists when trying to display multiple streams of data. When confronted with limited space and multiple streams of data, finding a way to effectively manage these various streams is non-trivial. Additionally, once all data has been displayed, the user is burdened with trying to discern which streams contain interesting information that they should be investigating.

The current method of dealing with multiple streams often involves opening each stream in an individual window and using a window manager to handle them. While this can be useful, most traditional window managers require a significant degree of manual user manipulation that makes them unwieldy and unscalable for large numbers of feeds. As identified by Kandogan and Shneiderman in the Elastic Windows paper, manual user interaction with a window manager adds a significant overhead to task completion [7]. The user has to discern which stream has interesting information, then drill down into that data while occluding other data.

Several dynamic window management techniques have been discussed in literature, however since the window manager is agnostic to the information being displayed in the windows, it is difficult for them to decide which fields are important and why they deserve the user's attention. By limiting our domain to Twitter feeds, our goal is to design a more efficient feed management system that can make interesting information readily apparent to the user.

We present TRENDTRACKER 1, a system to deal with multiple dynamic data streams and focus on interesting trends within those streams. TRENDTRACKER presents the user with multiple windows in one browser screen, each window containing Twitter trend information. The windows resize according to the rate at which that trend is being tweeted at that moment. Windows quickly grow and shrink according to which trend is currently being tweeted the most. We chose twitter trends as our data domain both for their quick rate of change and ease of access.

TRENDTRACKER is a completely automatic system that does not require, although allows, user manipulation. It creates a captivating system for the user to monitor multiple dynamic data streams and attract the user's attention to important data by actively monitoring feeds and changing window sizes. This allows the user to pick out the most popular trends as they are actively being tweeted.

II. RELATED WORK

TRENDTRACKER's main purpose to is effectively display multiple dynamic feeds of data. To do this it requires a few methods that focus on different domains. The first method is the use of an effective windowing algorithm to create a window for each feed and to autonomously manage the positioning and layout of the feeds within the confines of the web browser window. The second method deals with obtaining and processing the dynamic streams of data from Twitter. Below, we address current work in these areas.

Bell and Feiner's window management system described in their paper "Dynamic Space Management for User Interfaces" describes an algorithm that looks for the most efficient way to tile windows and manage empty space[2]. Their system provides a way to deal with a more typical desktop and user interface environment. Their paper provided us with a few ideas on how to go about implementing a windowing algorithm including representation of space as rectangles, adding a rectangle to a layout, and deleting a rectangle from the layout. Where Bell and Feiner decided to allow for empty space, TRENDTRACKER uses a tiling approach, allowing windows to fill the entire browser screen and eliminating any unused space. This helps to minimize many of the cases for which Bell and Feiner have to account for in their system. By scaling a box to have an importance relative to the overall space, we can resize boxes without having to worry about occlusion, overlap, or tiling. This simplifies the overall visualization implementation and appearance.

Another adaptive window management approach from which we draw is Miah and Alty's Vanishing Windows mechanism[9], in which windows are tiled and resized based on the degree with which the user interacts with them. Windows that have not been interacted with eventually are minimized and "vanish" from the screen. TRENDTRACKER dynamically resizes windows as well, however the size of a window is determined based on the rate of tweets rather than

on user interaction, allowing the system to function effectively display feeds even with a lack of user interaction.

While several other dynamic window management systems have been discussed in literature, including the Elastic Windows hierarchical approach [7] and the CUBRICON Intelligent Window Manager (CIWM) [5], our belief is that users have grown accustomed to having full control over their window management, making mainstream adoption of such algorithms difficult. Furthermore, many modern applications are have large toolbars, require large working areas, and do not scale well at lower resolutions, making a tiling system difficult. We address both of these issues. First, since our system is designed for passive monitoring rather than live interaction, dynamically resizing windows will not interfere with any of the users' operations. Secondly, because we design our windows to have minimal UI overhead and present only the title of the trending topic and the relevant tweets, our approach scales well, even at smaller screen resolutions.

There are a variety of applications and websites that have attempted to implement systems that allow users to track trends and stats surrounding Twitter, as this has been and continues to be a topic of interest. The data domain of Twitter is somewhat irrelevant, but its dynamic nature and categories of data interpretation are not. Our system provides information in the categories of trending topics, individual tweets, and changing in number of tweets per trend over the past few moments.

The following three related works of Tweetstats (Figure 2) [3], Trendistics (Figure 3) [4], and Monitter (Figure 4) [6] all provide a way to look at twitter trends. Tweetstats provides a word-cloud system that shows current trends and fifty most popular trends, with popularity encoded by size of the word. While word clouds are somewhat effective, Gestalt principles state the human perceptual system is limited in its ability to discern changes in area and associate them with a quantifiable value. Trendistics provides good stats about a specific trend and a good list of current trends. However, the system encodes information on a bar graph with percentages that lack context and change with different trends. Monitter provides a way to watch columns of trends. It allows for dynamic data to be updated into the trend and focus more on the tweets pertaining to the trend, rather than the variety of trends. These systems all provide different statistics for dealing with the dynamic nature of trends, however none are able to effectively display multiple trends at once and encode the rate of change for that trend at the same time. Rather the systems provide less information about multiple trends and cannot handle the large amount of data in the same manner.

The three systems also deal with individual tweets differently. Tweetstats does not provide any individual tweets. Trendistics provides the ability to display tweets for a specific trend, but requires refresh to update those tweets. Monitter allows for a similar experience that appears on the Twitter site itself, allowing a user to see the realtime results for a specific trend. Monitter updates each trend with tweets at the same rate however, giving the impression that each trend is as popular as the other. Neither of these systems are able to display realtime changes in the number of tweets for that trend. TRENDTRACKER provides a way to view the realtime results of a trend and the change in number of tweets for that trend. This helps to create context between the trends.

From these related works, we see the systems out there allow one to either drill down into a single piece of data, or look at large amounts of data. No system effectively combines the ability to deal with both at the same time. TRENDTRACKER provides a way to quickly see many trends and the data for that trend by combining a changing windowing algorithm not present in other systems.

III. METHODS

Determining the importance of a feed

The importance of a feed is determined by the rate at which new data is being presented to the user. By this mechanism, older, stagnating feeds are diminished from the user's view, while interesting topics with incoming data are brought to user's attention.

Window Layout and Placement

We considered a variety of approaches, including Bell and Fiener's overlap minimizing window management approach. We finally decided on a tiling algorithm that would have no overlap and fully utilize the entire screen. Unlike traditional tiling window managers such as Xmonad[11], in which a tiled layout is created by a sequence of horizontal and vertical subdivisions and then manually resized by the user, TRENDTRACKER uses an automated area-driven approach.

Funke describes advantages and disadvantages of a tiled system in his paper[5]. While a tiling algorithm allows all information to be fully visible, as windows do not get lost or covered, the size and number of windows is restricted by the screen area. Furthermore, it is difficult to size the windows to optimally fit information. We recognized these limitations when designing our system. To prevent windows from becoming too small and making information unreadable, we establish a hard limit on the minimum and maximum sizes of windows. Most importantly, tiling window managers are designed to allow fewer management operations from the user, which was the primary purpose of our system.

Our algorithm is implemented entirely in Javascript and works as follows. Importance values of each field are first normalized across fields by dividing the importance value of each feed by the total combined importance of all fields.

$$NormalizedImportance_f = \frac{Importance_f}{\sum Importance} \tag{1}$$

These normalized importance values now directly correspond to relative areas of fields in relation to the visible browser window. To allow for optimal placement of feeds and minimize any significant deviations in the aspect ratio beyond that of the browser window, our algorithm tries to place an equal number of rows and columns in the window. We start by finding the number of columns as the ceiling of the square root of the boxes.

$$|columns| = \lceil \sqrt{|boxes|} \rceil \tag{2}$$

We take the ceiling such that the number of columns will always be greater than or equal to the number of rows. Because the vast majority of standard displays are wider than they are tall, this prevents any significant deviations from a standard square-like aspect ratio in the fields. Next, we proceed by placing as close to an equal number of feeds in each column as possible. The height of each feed within the column is determined by the importance of that feed in relation to the total importance of the entire column, and the width of the column is determined by the importance of that column in relation to the combined importance of all the fields.

$$height_{feed} = \frac{importance_{feed}}{importance_{column}} * height_{screen} \quad (3)$$

$$width_{col} = \frac{importance_{col}}{\sum importance} * width_{screen} \quad (4)$$

In Figure 5, there is an initial box layout with exactly 7 fields, all of which are of equal importance and therefore equal area. As per the formulas, there are exactly 3 columns in which boxes are displayed.

In Figure 6, we have a box layout after window 1 (middle-top) experiences an increase in importance from 1.00 to 2.24 and window 3 (left-center) experiences an increase in importance from 1.00 to 1.50. Since importance values are normalized in relation to the total, the areas of the other boxes decrease to accommodate this change. Because the total importance of the left and center columns increase, the relative importance and width of the right column decreases.

In Figure 7, we have a box layout after two additional feeds, windows 7 and 8, are added to the stream and window 2's importance has shrunk from 1 to 0.4. When moused over, boxes change in color to red, allowing users to focus on fields of importance and visually separate a particular field of interest from the rest.

Animation

Changes in feed sizes and positions are smoothed by the use of animation. Abrupt movements of boxes without any smoothing detract from the user's attention and make it difficult to track individual feeds without forcing the user to refocus her attention each time boxes are moved.

Initially, we implemented animation by linearly interpolating changes in size and position at a constant pace each time importance values change. We attempted modifying this to use Jquery's animation API to smooth the transition by gradually accelerating and decelerating instead of moving at a constant velocity. However we found that as a result, when two adjacent feeds were resizing and when one needed to be resized to a greater degree than the other, the feeds would overlap during certain parts of the animation, due to uneven velocities causing one feed to reach a certain point faster than the other. As a result, we switched back to constant-velocity interpolation, which allowed all the feeds on the screen to move together at the same velocity and provided a much more pleasing visual effect. According to Lok and Feiner, a visual layout that is pleasing has a large impact on how well it communicates with those who are interacting with it [8].

IMPLEMENTATION

Pulling Twitter Feeds

We collect our trend data through the Twitter Streaming API[5]. Our application collects and queues data at a "Garden-hose" level, a sampling of public tweets that averages to 15% of the full public data that Twitter experiences. Through the use of a PHP implementation of the API called Phirehose[6], a script collects the JSON encoded streaming information and writes it to disk as it arrives, rotating file output every 5 seconds. Simultaneously, a consumption script reads these files and compiles the pertinent trend information.

The consumption routine reads through every tweet looking for trend information - a single word preceded by a hash tag (#). If It finds a trend, it updates the the internal list of trending data. It finds the trend in the trends list, or adds if it does not exist, and then increases its trend score by a constant amount of 250 points. Then, for every trend in our internal representation that was not mentioned in the current tweet, the trend score is reduced by one point. Trends that have reached a score of zero points are then pruned from the trends list.

The amount of score to add to a trend for each occurrence is based on the amount of incoming tweet data that the program experiences. If too few points are added for each occurrence, even popular trends can be represented as dying out very quickly. If too many points are added, even the least active trends stick around for too long before being pruned. The rate at which our program collected Twitter data is approximately 60 tweets/sec. At this rate, the addition of 250 points for each trend occurrence created a pleasing balance.

Determine Box Importances

The importance of a box (jQuery window) is determined based on the score of a trend. If the score of the trend increased from the last time Twitter was polled, then the box importance is increased by a constant amount of 1.3, or if the trend score decreased then the importance of the window is decreased by a constant amount of 1.3. The score for a trend is calculated based of the number of tweets that were tweeted for the trend over the period of time since the trend was last polled. By increasing and decreasing our box importance, the size of the box grows and shrinks accordingly. The speed of box growth and shrink can be adjusted by increasing or decreasing the value of 1.3.

Due to our windowing algorithm, it is possible to continually increase and decrease importance of boxes to the point where certain boxes completely occlude other boxes. To counteract this, we have implemented a max box and min box size. This keeps boxes from disappearing or from taking over the available window space. The max and min size is determined by box importance. Rather than giving the window an absolute area size, we allow the box to reach to a certain importance and then do not allow it move out of that threshold. We are essentially bounding the potential importance of boxes on both ends of the scale.

Displaying updates/incoming tweets

New trends and tweets are pulled in through the twitter phirehose into a temporary file. This file is then parsed into trends and their corresponding tweets. Those trends are then assigned to windows based on their scores, such that higher scores are put in lowered numbered boxes than higher number boxes. The trends' file is then updated with new scores and tweets. The new scores are then used to update the window importance. This scales the window accordingly. The new tweets can then be viewed by mousing over a box and watching new tweets appear at the bottom of the browser window. This helps to keep the visualization from altering too much, but also allow the user to view the most up-to-date tweets for a certain trend.

IV. Results

TrendTracker is useful for investigating twitter trends. It allows a user to carefully track twitter trends as they are changing and gain insight into what trends are popular, the tweets related to the trends, and the degree to which the trend is being talked about. We offer a scenario to help further elaborate on the type of analysis and information that is provided by TrendTracker.

Scenario 1: Marketing Information Analysis

A marketing agent for a large advertising firm has been instructed to figure out a new strategy for a product. The strategy needs to be in tune with what people are talking about and interested in. To begin with the agent begins by looking at typical surveys that are provided that help to detail what people are buying, doing, etc. However, the agent finds that this information is too routine and is already outdated by the time it reaches their desk. The agent continues to think of ways to tap into the market's pulse. They decide that a great new service that provides instant microblogging, Twitter, might be a great way to do just that. The agent is concerned about how to find large amounts of information about what people are talking about. The agent begins to investigate trending topics, but can't figure out a way to quickly understand what is being talked about at that moment. At this moment, the agent investigates TrendTracker and realizes it provides him with a plethora of up-to-date information surrounding peoples thoughts about multiple topics at the same time. Additionally, the system is picking out the most talked about trends, rather than the agent having to do this work. Armed with this new information and tool, the agent can craft the new strategy and ad campaign with more ease and understanding than ever before. The agent can quickly see peoples unfiltered thoughts about a subject and what their impressions about something are. It allows for a new way of brainstorming and advertising development that can be more targeted and in-touch with the pulse of the market.

Summary

The above example illustrates some key points about our system. TrendTracker provides trend information quickly, but more importantly it makes it easy for the user to find the important information quickly. It saves the user time by removing a large amount of mental processing that would otherwise be required. TrendTracker allows for the display of a large amount of dynamic data all at one time in a small amount of area. It is completely automatic, which allows the user to focus on the data and its analysis, and not the operation of the system.

V. Discussion

Our system has explored three main points: 1. Dynamic data should be displayed dynamically. The user can then receive constant feedback from the application and better understand the changing nature of the information. The challenge to overcome is in displaying the information in a way that makes sense to the user and in not changing on-screen elements too abruptly or quickly for a user to follow. 2. Draw the user's attention to the most important features. If there exists or can be devised a quantitative measure of importance for the various dynamic information being displayed, it is essential that the user's attention is guided to the most salient features. For a quickly changing dataset this is especially important as the user will have less time to evaluate on her own the features of the visualization. Methods of indicating importance can be any quantitative comparison element such as size, value, or relative motion. 3. Increase immersion by maximizing screen real estate usage. Our application fills the entire browser window and minimizes whitespace. When the high amount of information being streamed through application, judiciously and clearly increasing real estate usage increases the bandwidth of information passed on to the user.

VI. Future Work

Clustering and Coloring

Currently, our system places fields on the screen in arbitrary locations with no intelligent processing on the content of the tweets themselves. A useful addition would be to encode color into the category of the tweets, and to cluster similar tweets together based on their category, such as ”celebrities” or ”current events.” The category of a tweet could be determined from a cross-lookup on Wikipedia or Google Directory. Due to the wild variety of types of tweets, this makes it significantly easier for the user to focus on tweets of a certain category/color while filtering other information. Furthermore, users can have the option to click on tweets of a particular category to show only tweets from that category, which can in turn be broken into subcategories. For example, clicking on the ”celebrity” category would only show trending topics pertaining to celebrities and further subdivide into ”musician” and ”actor” subcategories.

Zooming

A known limitation with a tiling approach is a hard limit on the number of fields that can simulataneously be displayed on the screen. When displaying too many feeds in a constrained two-dimensional space, we encounter clutter. A potential solution to this would be to create a user interface

for zooming and panning through twitter feeds, similar to that used in Benderson and Hollan's Pad++ [1]. Of course, when zooming in on a particular region, the user loses visibility of all other regions that he may be interested in monitoring. However when coupled with the aforementioned "clustering" approach, showing a greater level of detail as the user zoom in, this could be a powerful mechanism for allowing users to literally "zoom-in" on topics of interest. Another interesting approach for increasing the number of feeds that can be displayed on the screen comes from utilizing depth. "The Task Gallery," is an example of such a system that uses a 3D metaphor to manage multiple windows in a virtual space [10]. A similar approach could be employed in TRENDTRACKER's tiling mechanism by applying a perspective transform to all the feeds, placing windows of higher importance closer to the user's viewpoint and placing less important windows closer to the vanishing point on the horizon. While this could potentially increase clutter, by utilizing a third dimension, this allows for displaying a larger number of feeds in a user's workspace.

REFERENCES

[1] Benjamin B. Bederson and James D. Hollan. Pad++: a zooming graphical interface for exploring alternate interface physics. In *UIST '94: Proceedings of the 7th annual ACM symposium on User interface software and technology*, pages 17–26, New York, NY, USA, 1994. ACM.

[2] Blaine A. Bell and Steven K. Feiner. Dynamic space management for user interfaces. In *UIST '00: Proceedings of the 13th annual ACM symposium on User interface software and technology*, pages 239–248, New York, NY, USA, 2000. ACM.

[3] Damon Cortesi. Tweetstats trends. http://tweetstats.com/, May 2010.

[4] Flaptor. Trendistic. http://trendistic.com/, May 2010.

[5] Douglas J. Funke, Jeannette G. Neal, and Rajendra D. Paul. An approach to intelligent automated window management. *International Journal of Man-Machine Studies*, 38(6):949 – 983, 1993.

[6] Alex Holt. Monitter. http://monitter.com/, May 2010.

[7] Eser Kandogan and Ben Shneiderman. Elastic windows: evaluation of multi-window operations. In *CHI '97: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 250–257, New York, NY, USA, 1997. ACM.

[8] Simon Lok, Steven Feiner, and Gary Ngai. Evaluation of visual balance for automated layout, 2004.

[9] Tunu Miah and James L. Alty. Vanishing windows: an empirical study of adaptive window management. In *Proceedings of the third international conference on Computer-aided design of user interfaces*, pages 171–184, Norwell, MA, USA, 1999. Kluwer Academic Publishers.

[10] George Robertson, Maarten van Dantzich, Daniel Robbins, Mary Czerwinski, Ken Hinckley, Kirsten Risden, David Thiel, and Vadim Gorokhovsky. The task gallery: a 3d window manager. In *CHI '00: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 494–501, New York, NY, USA, 2000. ACM.

[11] Don Stewart and Spencer Sjanssen. Xmonad. In *Haskell '07: Proceedings of the ACM SIGPLAN workshop on Haskell workshop*, pages 119–119, New York, NY, USA, 2007. ACM.

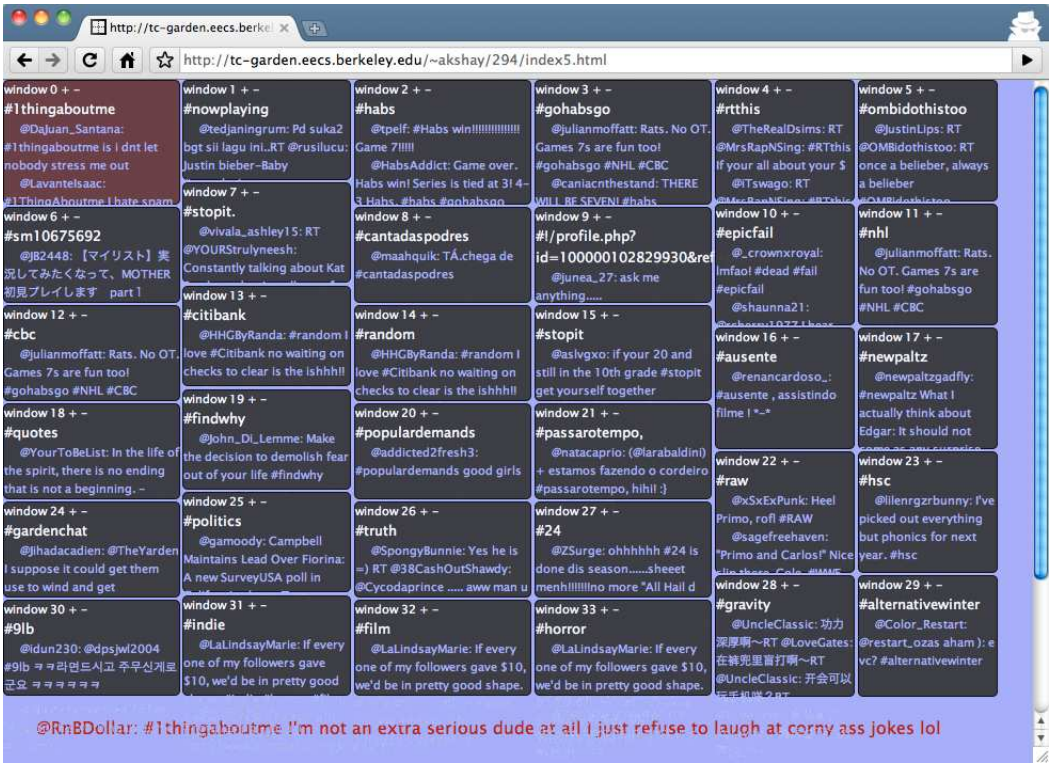

Fig. 1. TrendTracker - The system in action. The windows resize automatically. When mousing over a window you get new a red color to help with focus and new tweets that are arriving appear at the bottom of the page.

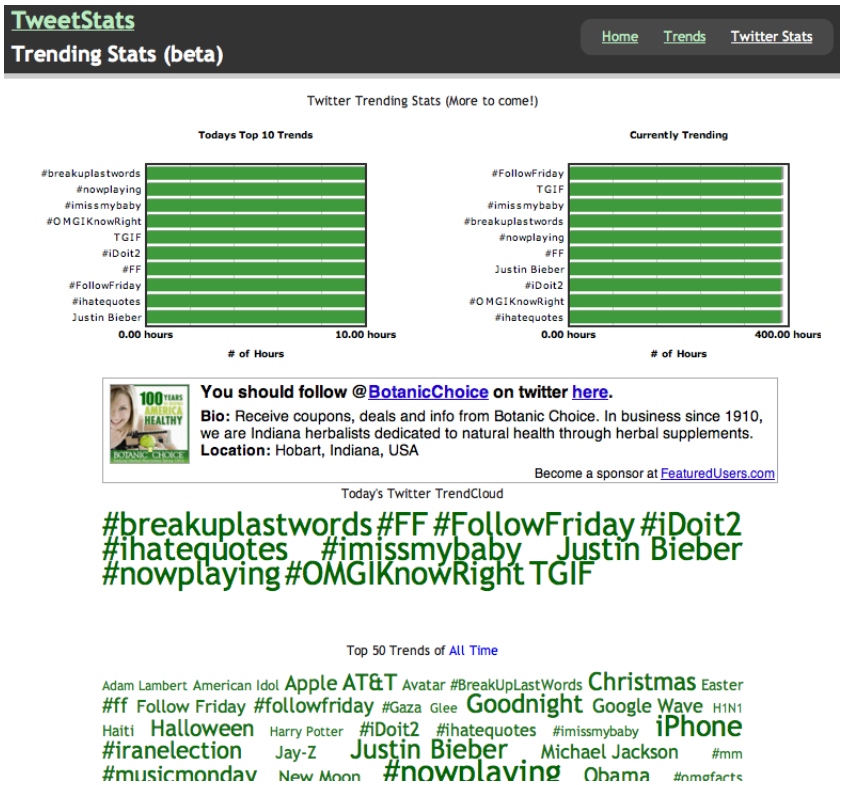

Fig. 2. TweetStats - A current implementation of twitter trend statistics. This system focuses on histories and uses a word cloud with word size difference to encode the popularity of the trend. However current trending topics has same size words.

Fig. 3. Trendistic - A current implementation of twitter trend statistics. This system focuses on a single trend and plotting its pertinent tweets and statistics. It does not update automatically (requires refresh). It lists other popular trends in a box on the right side of the screen.

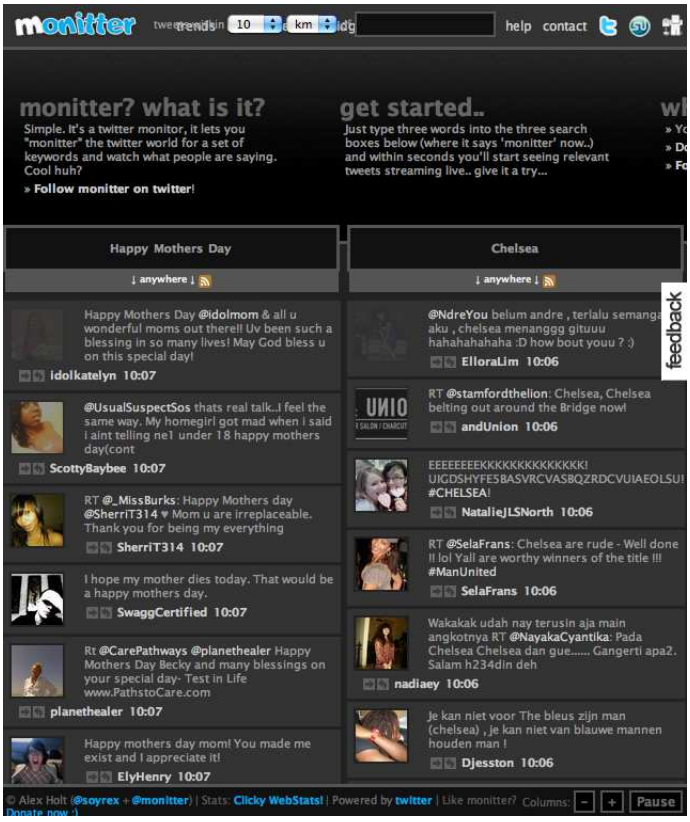

Fig. 4. Monitter - A current implementation of twitter trend statistics. This system provides a very similar interface to the one available on the twitter website. Its advantage is allowing the user to view multiple trends at once. However, trends update tweets at the same time, giving an illusion of equal popularity.

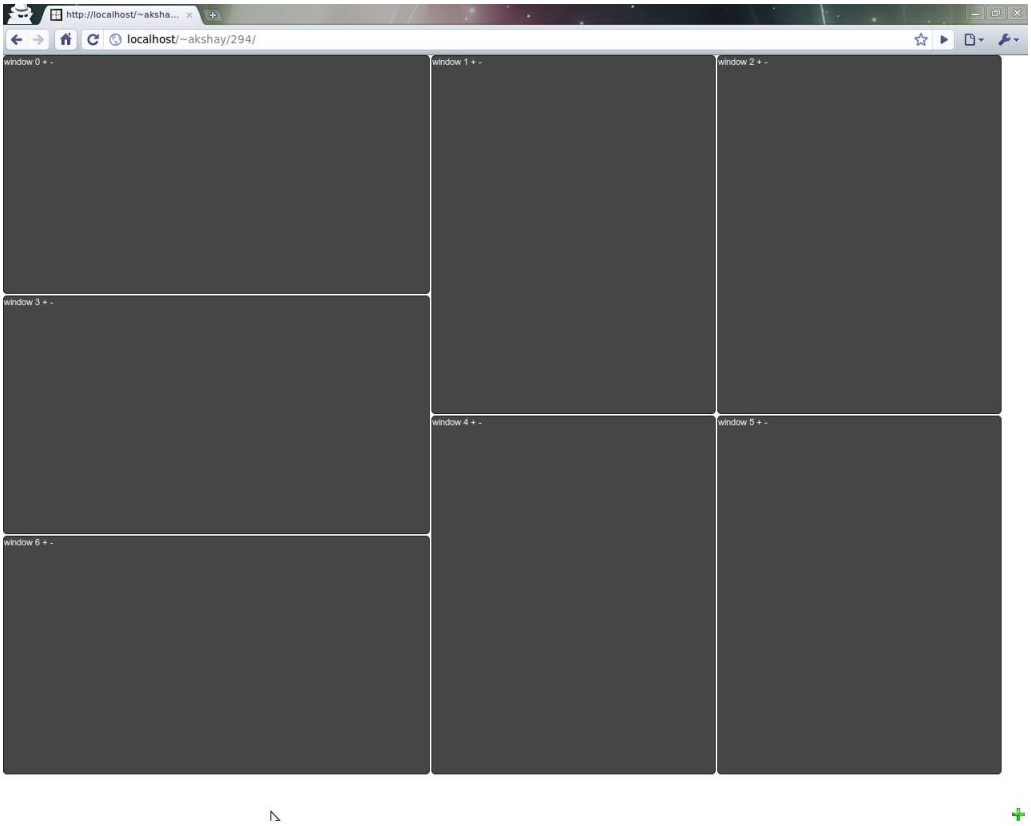

Fig. 5. Initial box layout with exactly 7 fields, all of which are of equal importance and equal area

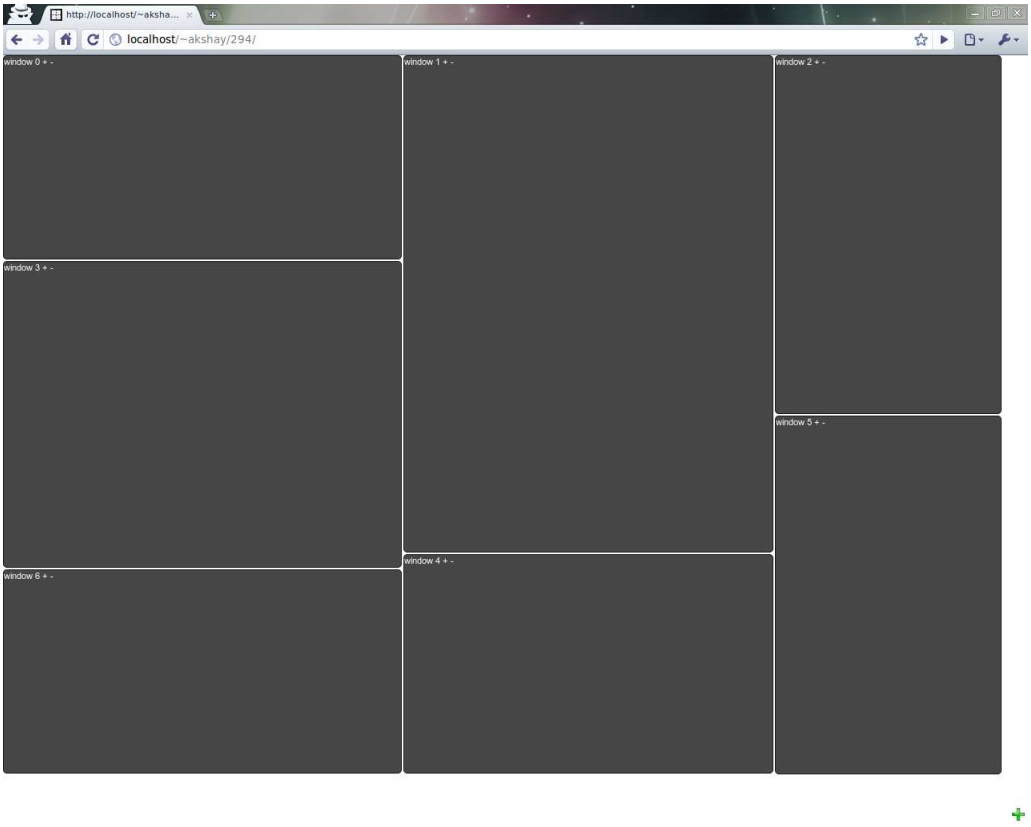

Fig. 6. Box layout after window 1 (middle-top) experiences an increase in importance from 1.00 to 2.24 and window 3 (left-center) experiences an increase in importance from 1.00 to 1.50.

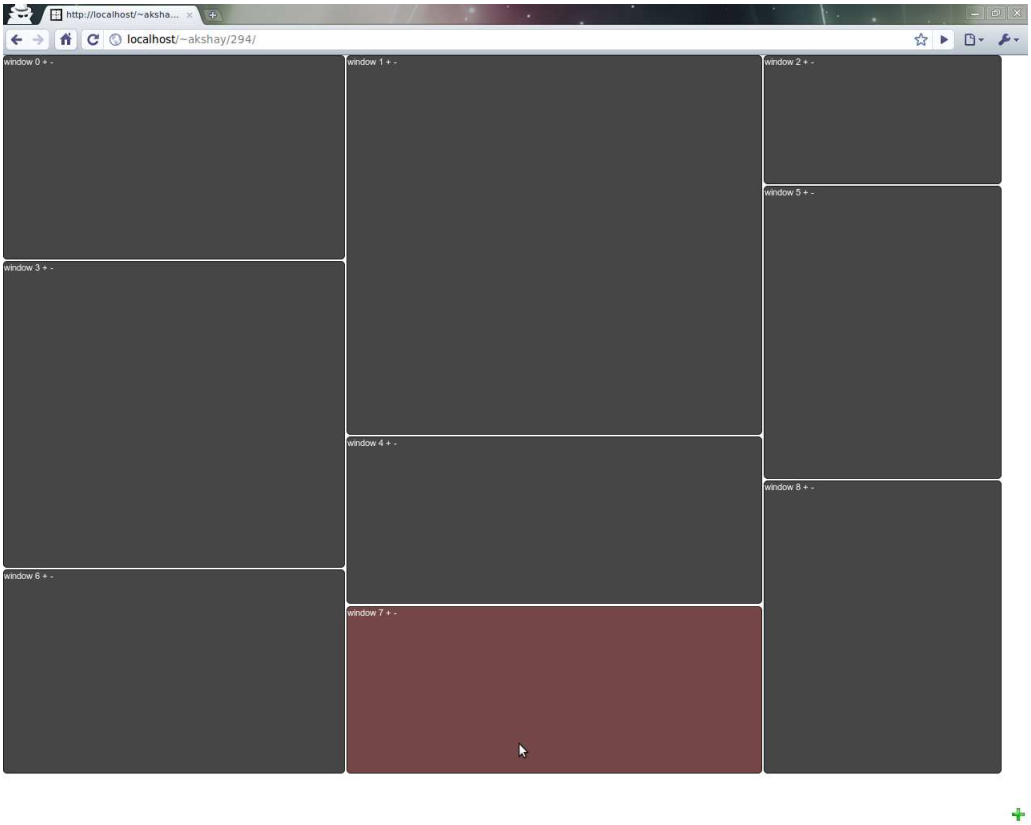

Fig. 7. Box layout after two additional feeds have been added. Window 7 and 8 have been added, causing window 2's importance to decrease from 1 to 0.4. When moused over the boxes change color to red, helping the user to focus on the field.