PIVE: A Per-Iteration Visualization Environment for Supporting Real-time Interactions with Computational Methods

Jaegul Choo, Changhyun Lee, and Haesun Park

Abstract—Visual analytics has been gaining increasing interest due to its fascinating characteristic that leverages both humans' visual perception and the power of computing. Although various computational methods are being proposed, they do not properly support visual analytics. One of the biggest obstacles towards their real-time visual analytic integration is their high computational complexity. As a way to tackle this problem, this paper presents PIVE, a Per-Iteration Visualization Environment for supporting real-time interactive visualization with computational methods. The main idea behind PIVE is that most advanced computational methods work by refining the solution iteratively. By visually delivering the result from each iteration to users, the proposed framework enables users to quickly acquire the information that the computational method provides as well as the ability to perform continuous interactions with them in real time. We show the effectiveness of PIVE in terms of real-time visualization and interaction capabilities by customizing various dimension reduction methods such as principal component analysis, multidimensional scaling, and t-distributed stochastic neighborhood embedding, and clustering methods such as k-means and latent Dirichlet allocation.

Index Terms—real time, interaction, visualization, multi-threading, clustering, dimension reduction, visual analytics

1 Introduction

The innate ability of humans to quickly perceive insight through visual analysis and decision processes has been a key factor in the growth of visual analytic research [17, 25]. One of the most significant efforts made by visual analytics researchers is the integration of various computational methods from data mining and machine learning areas with visual analytics so that users can benefit from intelligent meaningful information generated by these techniques. For example, dimension reduction and clustering methods have been commonly used in high-dimensional data visual analytics [5, 23]. More recently, latent Dirichlet allocation (LDA) [4], a popular method for document topic modeling, has been adopted in a wide variety of visual analytics systems for document analysis [28, 9, 18].

However, a critical hurdle in the integration of computational methods into visual analytics is the significant amount of computational time required by these methods. As computational methods become more advanced and capable, they usually run much slower, making it almost impossible to visualize and interact with them smoothly in real-time visual analytics. Due to this significant running time, even though numerous computational methods are currently being developed and some methods such as t-distributed stochastic neighbor embedding (t-SNE) [27] even claim their suitability directly in visualization applications, the state-of-the-art in visual analytics does not seem to fully utilize the advancements in computational methods. Consequently, in many domain areas, people still resort to only a few basic computational methods such as principal component analysis (PCA) [15] and multidimensional scaling (MDS) [7] for dimension reduction, hierarchical clustering and *k*-means [3] for clustering, etc.

However, we believe that various important aspects have been largely ignored when integrating (advanced) computational methods into visual analytics. In a sense that such an integration essentially involves both humans and computational methods, exploiting the characteristics of each side simultaneously may bring a synergetic effect for their tight integration that would not be possible otherwise. Motivated by this general idea, this paper focuses on the following aspects from each side: (1) *humans' perceptual precision* and (2) the *iteration-wise behavior of computational methods*.

For *humans' perceptual precision*, we highlight that when perceiving numbers, *humans do not require a high precision* such as a double or a single precisions typically used in modern computers. For example, when perceiving the value of π, most people know its approximate value, e.g., 3.14. In practice, perceiving it as a more accurate value, e.g., 3.1415926, does not make much difference . In a more analytic context, suppose the topic modeling has given a topic-wise representation of a particular document as (55.5852%, 38.8615%, 5.533%) with respect to three topics, e.g., science, sports, and economics. People may perceive its topic contribution at a tenth value at most, which is approximately (55.6%, 38.9%, 5.5%), but it would not change their perception significantly even if more accurate numbers were considered.

This substantially low perceptual precision compared to that of computational methods opens up a variety of possibilities to reduce the intensive computational time taken in running a computational method in a visual analytics environment. As a complementary characteristics of the computational methods to achieve this goal, we focus on their *iteration-wise behavior*. These days, many modern computational methods are performed through an iterative refinement process until reaching the final converged solution. An important observation found in most methods is that throughout the iterations, *a major refinement of the solution typically occurs in early iterations while only minor changes occur in the later iterations*. It indicates that the low-precision outputs of computational methods are dominated by their major refinement made during early iterations. In this respect, humans may be able to obtain most information from the computational method outputs in a much shorter amount of time than the full iterations until convergence.

However, apart from well-principled convergence criteria studied in most computational methods, it is not straightforward to determine when to terminate the iteration at which the result is reasonably accurate from the perspective of humans' perceptual precision. Instead, we propose an alternative approach called PIVE (**P**er-**I**teration **V**isualization **E**nvironment for supporting real-time interactive visualization with computational methods), which *visualizes the intermediate result per iteration as soon as they become available*. Unlike the previous approaches, which typically treat a particular computational method as a black box, the main novelty of PIVE lies in the idea to *break computational methods down to the iteration level and tightly integrate it with the interactive visualization so that users can check the result of computational methods without any delays and interact with them in real-time.*

- *Jaegul Choo, Haesun Park are with Georgia Institute of Technology. E-mail: {joyfull, hpark}@cc.gatech.edu.*
- *Changhyun Lee is with Georgia Institute of Technology. E-mail: clee407@gatech.edu*

Manuscript received 31 March 2013; accepted 1 August 2013; posted online 13 October 2013; mailed on 27 September 2013.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org.

Such real-time interaction capabilities based on this tight integration of computational methods at an iteration level makes significant differences in terms of the approaches for handling how we interact with a computational method. That is, from a perspective of viewing it as a black box, the turn-around time required for a particular interaction is usually equivalent to the time taken in running the entire set of iterations until its convergence. Therefore, previous efforts in adding an interaction capability to a computational method interactive have mainly focused on the sophisticated algorithmic modifications that can maximally reflect the users' intention from a single interaction. Accordingly, during a single interaction, it was generally recommended that users give the computational method a substantial amount of changes that are carefully made. Otherwise, users would be frustrated if the result due to a user interaction does not properly reflect their intention after a long time of waiting for the computational method to converge. On the contrary, in PIVE, a turn-around time for a single user interaction drastically decreases to the time taken in running a single or a small number of iterations at most instead of an entire set of iterations. In this respect, PIVE enables users to *perform multiple small interactions continuously by quickly adjusting their interactions based on the real-time response of the computational method*.

Motivated by these ideas, this paper discusses about PIVE in detail and present the example realizations of various well-known computational methods under PIVE. The main contributions of this paper is summarized as follows:

- Presentation of PIVE as a general idea to tightly integrate computational methods in visual analytics at an iteration level.
- In-depth discussion about the potential issues and their solutions in PIVE
- Realizations of PIVE with various well-known computational methods (PCA, MDS, t-SNE, *k*-means, and LDA) in established visual analytics systems
- Customizations of the above methods for real-time user interaction capabilities under PIVE
- Use cases of the customized methods with real-time user interaction examples

The rest of this paper is organized as follows. Section 2 discusses related work. Section 3 describes PIVE in more detail and discuss its potential issues and their solutions. Section 4 presents various customized computational methods with their supported interactions in PIVE. Using these customized methods, Section 5 describes the quantitative analyses about the iteration-wise behavior of computational methods and provide several use cases of the customized methods with their real-time interactions under PIVE in several well-known visual analytics systems. nally, Section 6 concludes the paper and discusses about the future work.

2 Related Work

In this section, we briefly discuss various previous studies from the two main perspectives: those aiming at efficient interactive visualization and those trying to make computational method user-interactive in visualization applications.

2.1 Efficient Interactive Visualization

Not surprisingly, numerous studies have focus on the visualization applications of large-scale data. Among various approaches, one of the straightforward but reasonable approaches is by using a subset of data by sampling. For example, Fisher et al. [13] has proposed an efficient way of dealing with large-scale data visualization by initially using only a small portion of data and then perform an incremental update on the visualization. Ellis et al. [10] has also taken a random sampling-based visualization approach mainly for avoiding the visualization clutter due to a large number of visualized objects while considering the efficiency issues during visualization.

As another popular approach for improve the efficiency in visualizing large-scale data, numerous studies have been based on multi-threading techniques. In this context, the main role of multi-threading is to separate the data processing/computation module and the visualization/rendering modules as multi-threads, allowing their efficient concurrent running. A notable line of research is called 'in situ' visualization [19, 32]. The main idea of it is, given large-scale data, to alleviate some post-processing overheads that had to be taken care of by the visualization module and let these overheads handled in the phase of the data processing/computation in which the powerful computing resource is readily available. In this manner, even though the visualization module does not have a computing power, which is often the case, the visualization can fluidly be performed. Although similar to the 'in situ' visualization approach, Tu et at. [26] has utilized the data sharing aspects in a parallel supercomputing environment. On the other hand, there have been approaches that have utilized multi-threading mainly for the purpose of providing a efficient responsive user interactions [21] by separating an application and a visualization threads into multiple concurrent threads.

As will described in detail in Section 3, PIVE adopts a similar multi-threading idea in order to reduce the overhead of the visualization module that has to go through a constant updating as the iterations of the computation method go. However, *none of these multi-threading-based approaches hardly exploited the nature of the iterative refinement processes found in most computational methods, which makes a clear distinction of PIVE to the previous work.*

Furthermore, efficient interactive visualization has been a main concern in the context of dynamic/streaming data. When visualizing dynamic/streaming data, the overall theme found in various approaches is to update the visualization efficiently given incremental changes in a data set. In this context, Cottam et al. [6] has recently discussed about a taxonomy for dynamic data visualization. Although the detailed approaches may differ, several prior studies [30, 31] have started from a relatively similar idea that the visualization update is carried out only when significant changes/events have been detected. Additionally, Alsakran et al. [1] has visualized the streaming documents using a GPU-accelerated force-directed layout technique.

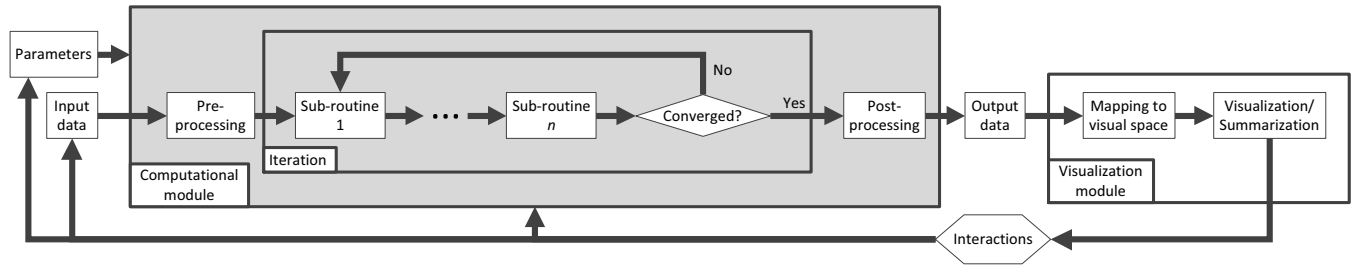
Various interesting ideas from dynamic data visualization could be applied to further improve the updating process of visualization in PIVE. Nevertheless, *the primary problem that PIVE tackles arises from the intensive amount of computations in the computational methods, and thus an efficient updating of the visualization module is not a concern in general.*

2.2 User Interaction with Computational Methods

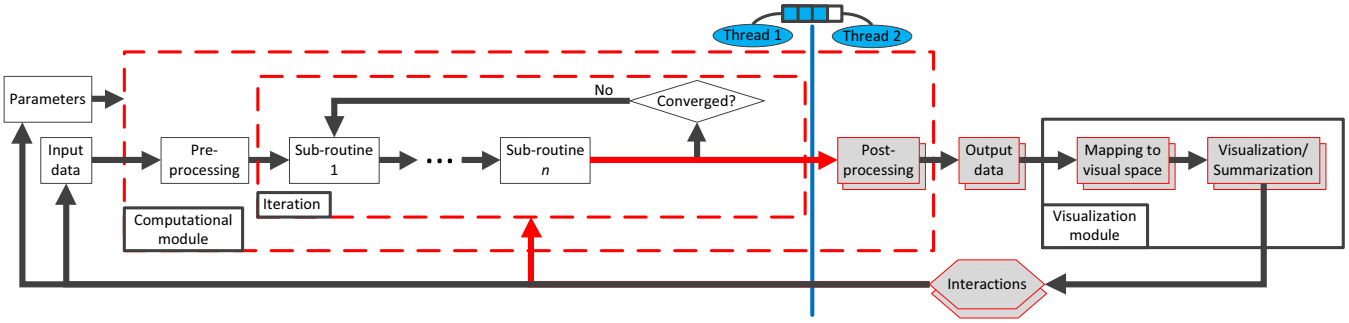
There have been numerous efforts to make computational methods, which are mostly automated, user-interactive in visualization applications. One of the most representative work is based on MDS [29] that has added MDS a capability of incorporating user feedback based on a user-specified visual region. A more recent work called observation-level interaction [11] has provided a general framework in which the user interaction from a scatter plot is incorporated in a Bayesian probabilistic framework.

These user interaction capabilities have long been emphasized in terms of clustering since clustering is generally a difficult problem. Seo et al. [23] has improved a traditional clustering method called hierarchical clustering so that it can have flexible interactive capability with the clustering result in a bioinformatics domain. More recently, iVisClustering [18] has tried to make a more recent method LDA interactive by supporting cluster merging/splitting, cluster keyword refinement, etc.

However, most methods have treated the computational method as a black box, and thus the interactions they support are inherently far from being real-time because the entire set of iterations for a new run of the computational method is required for each iteration. Nonetheless, a variety of work has addressed the importance of the capability for supporting a continuous set of real-time interactions with computational methods due to the highly exploratory nature of human interactions [12, 21, 24]. In this sense, PIVE, which leverages both humans'



(a) The standard workflow



(b) The PIVE workflow

Fig. 1: An overall diagram of PIVE (b) in contrast to the standard (non-iteration-wise) one (a). In the standard framework (a), a computional method is treated as a black box, as depicted by a gray rectangle. On the other hand, PIVE (b) breaks down the computational method at its iteration level, allowing it to be visualized at each iteration while taking into account any user iteractions. The blue line separates the overall procedure into two separate threads with their message queue, as shown in the blue rectangle, to remove potential computational overheads.

perceptual precision and the iteration-wise behavior of computational methods, bears a potentially great impact in achieving this goal.

3 Per-Iteration Visualization Environment (PIVE)

First, we describe an overall flow of PIVE (Fig. 1(a)), by highlighting its differences from the standard (non-iteration-wise) approach (Fig. 1(b)).

Let us begin with a general procedure when an iterative computational method is integrated in visual analytics. As shown in Fig. 1(a), input data, which are usually represented as multidimensional vectors, are given to the computational module along with its required parameter values. The computational module pre-processes the data, if necessary, and runs through iterations, which are usually divided into multiple sub-routines, until it converges. Upon convergence, the output goes through a post-processing step.

The final output of the computational module is then passed to the visualization module, which encodes it in a visual space and finally delivers its visualization to users. For example, the output of a dimension reduction method, e.g., PCA, can map data items onto the coordinates of the screen space, and the output of clustering can be used to color-code each group of data clusters.

Users can then better explore the visually represented data with the help of the information provided by the computational method and often interact with computational methods by adjusting their input data as well as their parameters. These interactions trigger another run of the computational method. For example, given the cluster summary for a set of text documents, if a user finds an interesting cluster, the user may perform another iteration of clustering on the particular subset to obtain more details about the chosen subset. On the other hand, users might want to adjust the number of clusters, which is usually a user-specified parameter in clustering methods, to find the best clustering result for the data.

In most of the described visualizations and interactions, the standard framework generally treats the computational module as a black box, which the visualization module has no control over, depicted by a gray rectangle in Fig. 1(a). In other words, once the computational module has been initiated, visual analytic systems must wait for it to finish its iterations before it outputs the visualization to users.

On the contrary, PIVE takes the results of intermediate iterations out of the computational module and delivers them to the visualization module whenever they are available. More specifically, as highlighted with the red horizontal line in Fig. 1(b), the result from each iteration is always passed to the post-processing step, the output of which, in turn, reaches all the way to the visualization module, regardless of whether it has converged or not. Consequently, these intermediate results are visualized to users much more quickly than having to wait for the converged solutions.

In addition, PIVE enables the above-discussed interactions to be instantly reflected by directly interacting with the process for each iteration of the computational module, as highlighted with the red vertical line in Fig. 1(b). For instance, given the result of a particular iteration, one could exclude certain data items from the following iterations, which accelerate the later iterations due to the reduced data size. Furthermore, users could change the number of clusters while a clustering method is running, which immediately affects the following iterations.

3.1 Issues and Solutions

3.1.1 Computational Overhead and Multi-threading

Computational overheads are one of the issues that can be potentially introduced by this framework. As can be seen in the red-lined stacked rectangle blocks in Fig. 1(b), visual analytics systems have to process the output for each iteration repetitively while the standard approach needs to process only the final output once. These additional computations could undermine the effectiveness of the proposed framework. Let us suppose that a particular computational method, which requires 50 iterations to converge, converges in a minute. If the proposed framework runs only 4-5 iterations within the same amount of time, then users might prefer the standard approach instead of being able to check the intermediate results since the results from such early iterations may not be satisfactory.

However, we claim that this issue can be easily overcome by applying a multi-threaded approach to the proposed framework. As shown by the blue ellipses in Fig. 1(b), the entire process can be separated into two concurrent processes/threads. The first thread shown to the left is responsible only for the sub-routines inside the iteration while the second thread on the right handles actions from the post-processing block to the visualization block. These two threads communicate with each other via a message queue, as shown by the blue rectangle on top in Fig. 1(b), where the outputs for each iteration for post-processing are to be stored.

Modern computers are usually equipped with at least two or more cores on the CPU. These two threads can be executed virtually in parallel, which hardly slow down the computational methods compared to the standard approach. Although not included in this paper, for the computational methods we customized, we compared the total computing time between PIVE and the standard frameworks, but with multi-threading implemented, there were essentially no differences in their running times.

Even in this multi-threading framework, the following case may still be problematic. Suppose the second thread involves more intensive computations than the first thread because, for example, the post-processing block takes more time than the processes at each iteration. As a result, the second thread would act as a bottleneck in the overall flow of the proposed framework, resulting in the message queue increasing. One way to handle this issue is to store the results of each iteration periodically rather than storing every one of them in the first thread. Alternatively, the second thread could take the most recent iteration-level results and discard the remaining older ones from the message queue. Under this situation, the visualization of the intermediate results may be somewhat discontinuous, but users would always be given the most recent result, which should be the most accurate solution up to the current iteration.

Finally, the other overhead comes from copying results from each iteration to the message queue, which results in a memory write operation. In the standard approach, these results for each iteration are usually written to the same memory space over iterations since the results from previous iterations do not need to be maintained. However, memory write operations are generally very fast. Furthermore, the outputs from each iteration of computational methods take up a much smaller memory compared to input data. For example, even if the data is a very high-dimensional, say, in the hundreds of thousands of dimensions, such as is the case in text data, the dimension reduction outputs would only be two-dimensional representations assuming they are visualized in a 2D space. Since the amount of additional computational time and memory that is required by our approach is minimal, we do not see memory overheads being a critical issue.

3.1.2 Visual Inconsistency and User Control

The second issue in the proposed framework is the visual inconsistency, which occurs during visualization updates, due to dynamic results changing each iteration. The most severe case occurs when the visualization changes too frequently. Although the amount of change generally diminishes as the iterations proceed, frequently changing visualizations may prevent users from obtaining a consistent picture of the data.

To address these issues with visual inconsistencies, we've come up with several possible controls. The first most basic option would be a stop and resume control which would stop and resume updates of the visualization. Secondly, a time period control would manage the length of the visualization. Additionally, we could pair this time controller with two choices - the option to visualize the most up-to-date result or to visualize the result of the next item in the queue, which would provide the user with smoother visual transitions. Similar to the 'stop/resume' interaction, since our approach maintains each of the intermediate results, we could simply expand the controls to also add both the 'play backwards' and 'jump to…' options. These interactions would help users understand the overall trajectory of the results through each iteration. Through the use of these controls, it is very possible that the user may uncover an interesting insight into the data at a particular iteration or a series of iterations.

4 Customized Methods under PIVE

In this section, following the proposed framework, we present several customized computational methods in visual analytics systems. To begin with, we have chosen three visual analytics systems, FodavaTestbed,[1] Jigsaw,[2] and iVisClustering [18], which involve computational methods.[3]

FodavaTestbed is a visual analytics system for high-dimensional data, where users can apply various dimension reduction and clustering methods for exploratory analysis. Among various methods supported, we have chosen three dimension reduction methods, 1. MDS, 2. PCA, and 3. t-SNE. Jigsaw is a well-known system for document analysis, and we have chosen 4. *k*-means, which is used to provide a summary in terms of a compact set of clusters. Finally, iVisClustering is an interactive document clustering system which uses 5. LDA, a popular topic modeling method.

In the following, we describe how each method is customized along with the additional interactions we implemented in the proposed framework.

4.1 Principal Component Analysis (PCA)

PCA [16] is a well-known dimension reduction method that captures the maximal variance in the data via a linear projection. PCA is mainly based on the method called eigendecomposition, the algorithms of which are categorized into two different methods, the QR algorithm and the Lanczos algorithm [14].

Basically, the Lanczos algorithm approximates a given data matrix by a much smaller one in the Krylov subspace [14], the dimension of which iteratively expands, and efficiently solves the eigendecomposition on the latter matrix. Due to the nature that this matrix well-approximates the largest eigenvectors of the original one, the Lanczos algorithm performs much faster than the QR algorithm in visual analytics in which only a few dimensions are needed.

We customize the Lanczos-based PCA implementation of FodavaTestbed so that the results for each iteration are dynamically visualized.

4.2 Multidimensional Scaling (MDS)

MDS [7] is a traditional dimension reduction method that attempts to preserve given distances/relationships of data items in a lower-dimensional space. Given the ideal distance δ_{ij} between x_i and x_j, MDS solves

$$\min_{x_1,\ldots,x_n} \sum_{1\le i<j\le n} \left(d_{ij}-\delta_{ij}\right)^2, \tag{1}$$

where d_{ij} is the distance between the reduced dimensional vectors x_i and x_j. A Euclidean distance $\|x_i - x_j\|_2$ is usually used for d_{ij}. Solving Eq. (1) iteratively refines x_i's based on various optimization techniques [8]. We customize MDS in FodavaTestbed by extracting the x_i's at each iteration from the MDS implementation.

4.2.1 User Interaction Capabilities

Additionally, while the results for each iteration of MDS are visualized in a scatter plot, we support the interaction capability that enables users to move the data points by mouse via drag-and-drop, similar to the Prefuse force-directed layout. Then, during the MDS iterations, their new positions in the screen space are translated back to the MDS output coordinates, x_i's. The changes in x_i's at a particular iteration then affect the following iterations by generating different d_{ij}'s. In terms of how MDS behaves due to these changes, we provide two different capabilities: 'soft' vs. 'hard' placement. The soft placement continues iterations without any changes in MDS behaviors. It is equivalent to restarting MDS with the intermediate result at the particular iteration as the initial values for x_i's.

[1] `http://fodava.gatech.edu/fodava-testbed-software`
[2] `http://www.cc.gatech.edu/gvu/ii/jigsaw/`
[3] We obtained the code from the original authors of the systems.

The hard placement capability fixes the values of x_i's for points moved by the user. This can be easily achieved by skipping the update step of these x_i's in the following iterations. Note that, however, even though their values do not change, other data points are still influenced by these fixed points, and in this sense, our approach is a semi-supervised MDS that reflects user interventions.

When using the semi-supervised MDS, an important advantage of the proposed framework is that users can immediately check the effects of these interactions via the iteration-wise visualization. Our modifications in FodavaTestbed support both types of interactions.

4.3 t-Distributed Stochastic Neighbor Embedding (t-SNE)

t-SNE [27] is a relatively new dimension reduction method. It interprets pairwise distances as probabilities both in high-dimensional and lower-dimensional spaces and tries to minimize their Kullback–Leibler divergence, a distance measure between probability distributions. Unlike the previous methods discussed, it focuses on preserving neighborhood relationships instead of global ones, and it has shown its outstanding capabilities in visualizations.[4]

Although we skip the detailed formulations because of the scope of the visual analytics community, the algorithm works iteratively by refining the lower-dimensional coordinates based on a gradient descent-based framework. In practice, however, t-SNE does not provide a clear stopping criterion, and thus it typically iterates several hundred times by default for any data set, which usually takes a significant amount of time. We customize the t-SNE in FodavaTestbed in a similar manner to the way we altered MDS.

4.3.1 User Interaction Capabilities

Likewise, we provide both the soft and hard placement interactions for t-SNE, as discussed in MDS. Although the algorithm details are different, the overall iterative procedure turns out to be quite similar to MDS. Thus, for the soft placement, we restart t-SNE with the intermediate results immediately during iterations. For the hard one, we skip the update step for data items moved by the user in the following iterations while they still influence other points in the t-SNE iterations. Therefore, our altered method can be viewed as a semi-supervised t-SNE.

4.4 *k*-means

k-means, which is a widely-used clustering method, performs the following steps iteratively: 1. computing the centroid of each cluster by averaging the data vectors in the corresponding cluster and 2. updating the cluster assignment of each data item based on its closest cluster centroid. The iteration terminates when there are no cluster membership changes.

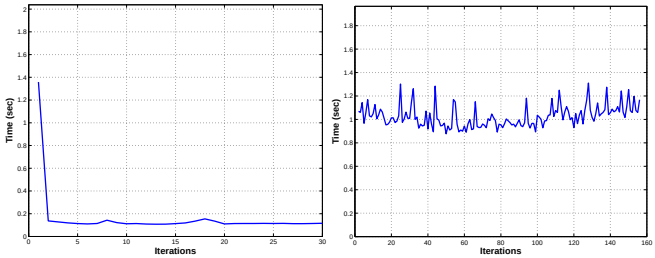
Although Jigsaw provides a cluster view based on *k*-means, it currently visualizes only the pre-computed results since *k*-means is usually very slow to converge. We customize it so that users can run *k*-means in real-time and the intermediate cluster memberships are dynamically visualized.

4.4.1 User Interaction Capabilities

Additionally, we add several interaction capabilities in the proposed framework. One is to split/merge clusters during iterations. On a split/merge interaction, similar to the soft placement in MDS and t-SNE, *k*-means restarts with the intermediate cluster memberships that reflect split/merged clusters, involving dynamic changes in a *k*-means parameter which represents the number of clusters.

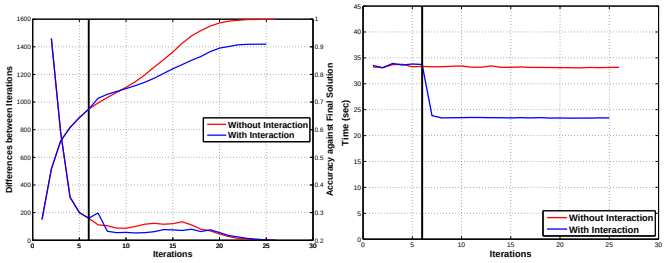
Similar to the hard placement in MDS and t-SNE, another capability we provide is the option to fix the cluster assignments of the data in a particular cluster. To accomplish this, we skip the updating step of the cluster assignment for these data in the following iterations. However, they still contribute to the centroid computing step. A similar semi-supervised way of *k*-means was previously proposed [2], but our framework significantly accelerates such interactions with *k*-means.

[4] `http://homepage.tudelft.nl/19j49/t-SNE.html`



(a) PCA (b) MDS

Fig. 3: The computing times of the example in Fig. 2.



(a) The cluster membership changes (b) The computing times

Fig. 5: The behaviors for each iteration of *k*-means with and without the interaction made in Fig. 7(b). In (a), the decreasing lines are the cluster membership changes between the current and the previous iterations while the increasing ones are the correct cluster memberships with respect to the final solutions without the interaction. The black vertical line represents the iteration point of the interaction made.

4.5 Latent Dirichlet Allocation (LDA)

LDA [4] is a popular topic modeling method for documents based on a generative probabilistic model. Given a number of topics, it gives two outputs: the term-wise distribution of each topic and the topic-wise distribution of each document. The iterations of LDA basically update these two outputs alternately. From a clustering viewpoint, the former corresponds to a centroid vector of each topic cluster, and the latter to a soft-clustering coefficient. By taking the topic index that has the maximum value in the latter, a document is clustered to a particular topic.

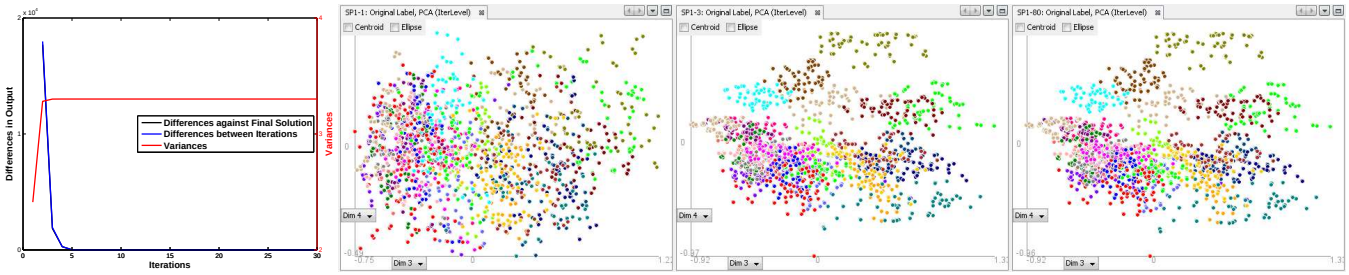
iVisClustering uses one of the fastest LDA libraries called Mallet [20], which implements LDA based on a Gibbs sampling [22]. Although this sampling-based approach does not guarantee a convergence, it is being widely used because of its simplicity and robustness against overfitting, compared to the variational approximation method proposed in [4]. Due to this convergence issue, LDA usually iterates a pre-defined number of iterations and usually requires a significant amount of time. We customize the Mallet library so that it can give the outputs from each iteration to iVisClustering, allowing iVisClustering to dynamically update its visualization.

4.5.1 User Interaction Capabilities

In addition to the original iVisClustering interaction capabilities being available during iterations, we also add several interactions with LDA in iVisClustering, similar to those in Jigsaw: splitting/merging clusters and fixing the cluster assignments of particular data items during iterations. The customization of LDA for such interactions is similar to *k*-means, and thus we skip the details due to the page limit.

5 Experiments

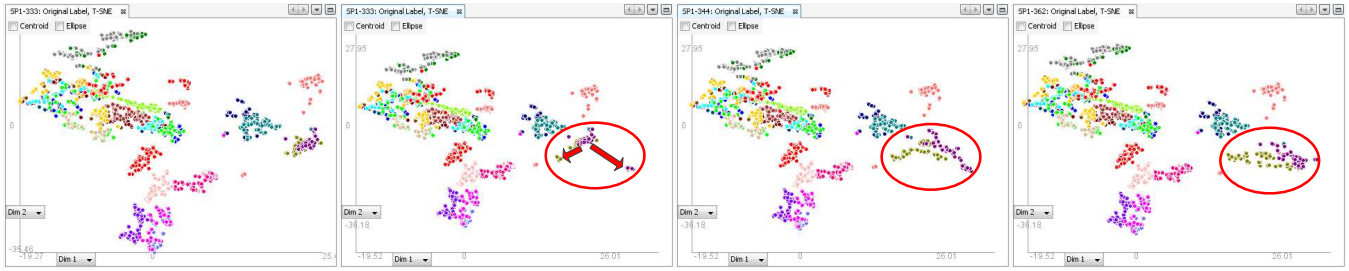
In this section, we present behaviors within each iteration for computational methods as well as their interactive aspects in the proposed framework.



(a) PCA criteria values and output changes (b) The scatter plot at the first iteration (c) The scatter plot at the third iteration (d) The scatter plot at the 80th iteration

(e) MDS criteria values and output changes (f) The scatter plot at the 10th iteration (g) The scatter plot at the 30th iteration (h) The scatter plot at the 156th (converged) iteration

Fig. 2: The behavior for each iteration of PCA and MDS and their visualization snapshots. In (a) and (e), the red lines represent the criteria values of each method, the lower-dimensional variance in PCA and the stress value, i.e., Eq. (1), in MDS. The blue line is the Euclidean distances of the lower-dimensional outputs between the current and the previous iterations, and the black line is those between the current and the final iterations. In (e), the black and the blue lines almost coincide. In PCA, 1,420 facial image data representing pixel values in 2,048 dimensions have been used, and in MDS, 500 handwritten digit data representing pen traces in 16 dimensions have been used.



(a) The 333th-iteration result (b) The 333th-iteration result after moving points (c) The 344th-iteration result (d) The 362th-iteration result

Fig. 4: A point-moving interaction example using t-SNE. The two overlapping clusters,'l' and 'o,' are separated due to a user interaction of moving apart a few points from each cluster. 1,558 spoken letter data represented in 618 dimensions have been used.

5.1 Iteration-wise Behaviors and Visualization

Fig. 2 shows the behaviors of each iteration for the customized PCA and MDS along with their computing times shown in Fig. 3. In PCA, the criteria value, i.e., the lower-dimensional variance, as well as the lower-dimensional outputs (Fig. 2(a)) converge within a few iterations, indicating that only a few iterations of the Lanczos algorithm are needed in visual analytics applications (Figs. 2(b)-(d)). Nonetheless, each iteration takes roughly the same amount of computation time except for the first iteration which includes the pre-processing time. (Fig. 3(a)). Instead of having to perform a fairly large number of iterations, as most PCA algorithms do, the iteration-wise visualization enables users to obtain an equivalent visualization much quickly. A similar argument applies to MDS as well. Although its convergence is relatively slow compared to PCA (Fig. 2(e)), we obtain the results similar to the final one achieved at the 10th iteration (Figs. 2(f)-(h)). We do not present the quantitative analyses for t-SNE, but we found tendencies similar to MDS, and we will focus on its interactive aspects in the following section.

In clustering, the behavior of each iteration of *k*-means is presented in Fig. 5 as well as their snapshots in Jigsaw in Fig. 7. In Jigsaw, in order to best assist users in easily identifying the location and the number of changes that occur while the visualization is dynamically updated, we draw arrows to represent where a particular data item has moved relative to the previous iteration, and color-code each data item to represent which cluster index it previously belonged to. As shown in Fig. 7 and in the redlines in Fig. 5, although significant change occurs in early iterations (Fig. 7(a)), they diminishes quickly, as seen in the sixth iteration (Fig. 7(b)), which is not much different from the final result (Fig. 7(d)). However, the time each iteration takes is almost the same (Fig. 5(d)).

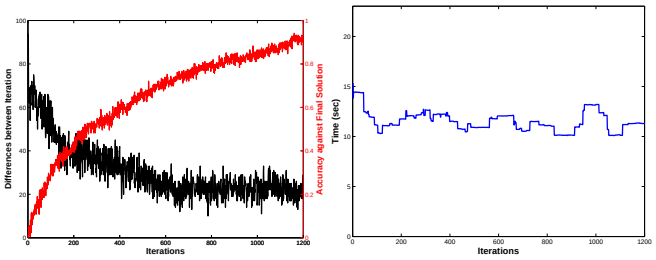
Finally, LDA, which is a sampling-based approach, shows a significantly different behavior from the previous methods (Fig. 6). Although cluster membership changes between iterations generally decrease and the solution narrows to the final solutions (Fig. 6(a)), cluster memberships change significantly even after many iterations, in this case after 1,200 iterations. In iVisClustering, we could see the top keywords of each topic become somewhat stable after several hundreds of iterations (Fig. 6(a)), but the randomness of the sampling-based algorithms might make it harder to give consistent visualizations when compared to deterministic methods in PIVE.

Table 1: The keyword summaries of the sampled clusters with/without fixing interactions of *k*-means performed in Fig. 7.

	Cluster 1	Cluster 6	Cluster 7	Cluster 8	Cluster 10
Fig. 7(c)	process,trying,latency	turing,100,budget	quasimonte,unbalanced,choice	concern,rich,solvable	intel,orchestrate,nonapproximability
Fig. 7(d)	schur,process,trying	turing,budget,100	quasimonte,unbalanced,choice	concern,rich,solvable	intel,orchestrate,nonapproximability

Table 2: The keyword summaries of the selected clusters during splitting and merging interactions of *k*-means performed in Fig. 8.

	Cluster 1				Cluster 2		Cluster 3	
	1-a	1-b	1-c	1-d	1-a	1-b	1-a	1-b
Fig. 8(a)	analysts,cdc, earliest	weird,contents, oneyearold	fundraising, 11,7bd	chromosomes,100fold, may605375rossignol	warm,rhythms, pretend	37,symptoms, said	social,cause,symptoms	
Fig. 8(b)	enjoy,contents,weird				warm,37,rhythms		social,causes, people	social,causes, cerebellum
Fig. 8(c)	enjoy,contents,weird				warm,37,symptoms		1000,social, causes	incredible,symptoms, cerebellum



(a) The cluster membership changes (b) The computing times

Fig. 6: The iteration-wise behaviors of LDA. In (a), the blue line represents cluster membership changes between the current and the previous iterations while the red line represents the correct cluster memberships with respect to the final solutions.

5.2 Real-time User Interactions

Basically, in all three systems, we provide basic interactions that control the visualization for each iteration, as discussed in Section 3.1. In the following, we show several use cases of the interactions discussed in Section 4.

5.2.1 Moving data points in t-SNE

Fig. 4 shows an interesting interaction which involves moving a data point in t-SNE. Given some overlapping clusters in a particular visualization generated by t-SNE (Fig. 4(a)), users place several points from different clusters far apart (Fig. 4(b)), and then t-SNE reflects these changes in the following iterations, resulting in tje separation of most points in two clusters from each other (Figs. 4(c)(d)). This simple, yet powerful example clearly illustrates the advantage of providing users with the ability to interact with computational methods in our framework in real-time visual analytics.

5.2.2 Fixing cluster assignments in *k*-means

For our *k*-means method, we provide users with another interaction that allows them to fix cluster assignments for particular data items at a certain iteration. This interaction becomes especially useful when users feel that particular clusters are adequate and want to prevent them from changing much. In addition, fixing some clusters that are already stable in early iterations can accelerate the later iterations by excluding them from the cluster assignment step.

Fig. 5 shows the effects of such interactions. First, we start with the same example shown in Fig. 7, but we fix the clustering assignments of the cluster highlighted in yellow rectangles, which amounts to 44% of the total data items, at the sixth iteration (Fig. 7(b)). Once this interaction is performed, the computing times for the following iterations of *k*-means drops significantly (Fig. 5(b)). However, only less than 10% of the final cluster memberships differ from the final results without this interaction, as shown in the increasing red line in Fig. 5(a). The final outputs of the cluster view in Jigsaw of the two cases can also be compared in Figs. 7(c) and (d), both of which are similar in terms of cluster sizes as well as keyword descriptions.

5.2.3 Split/merge clusters in *k*-means

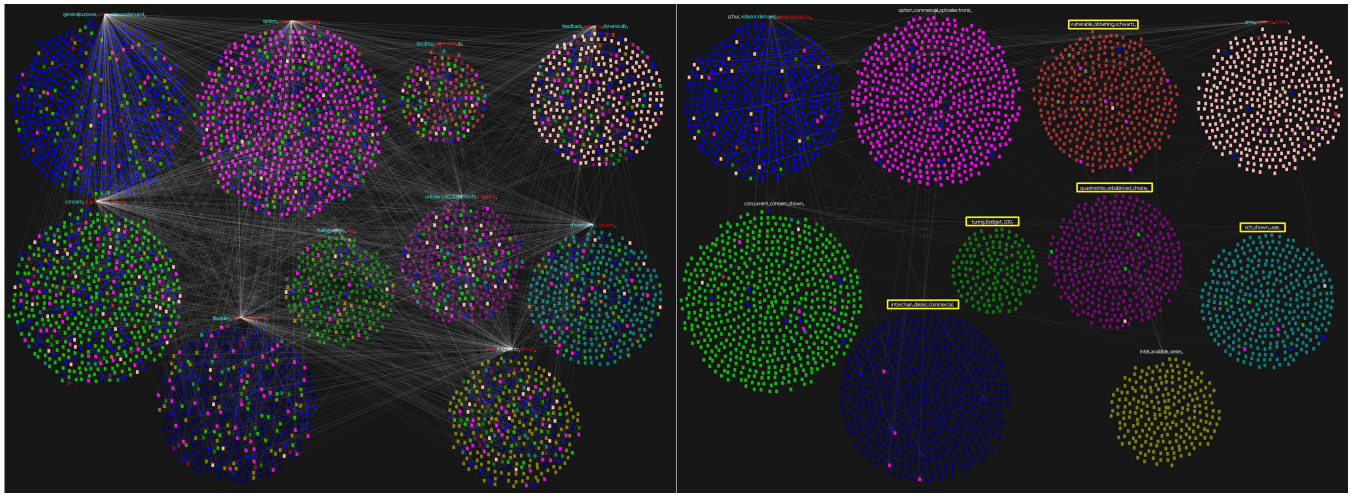
Our customization of *k*-means enables users to merge multiple small or semantically related clusters or split large or unclear clusters. Fig. 8 shows its example in Jigsaw. In the third iteration, we merge yellow and green clusters and split a white cluster (Fig. 8(a)). The resulting is shown in Fig. 8(b). We obtain a much more balanced set of clusters (Fig. 8(c)) compared to the final result in which no splitting/merging was performed (Fig. 8(d)). Furthermore, after analyzing the documents in two split clusters, we found that one of the clusters primarily contained documents about the causes of autism while the other about the symptoms, as seen in the keyword summary in Fig. 8(c). Without the interaction, one will notice in Fig. 8(d) that these clusters are not easily separated.

5.2.4 Filtering noisy documents to improve topics in LDA

The ability to filter noisy documents has been an appealing interaction for LDA in iVisClustering. To be specific, given parallel coordinate representations of topic-wise distributions of documents, users can interactively filter out documents that are not strongly related to a single topic, i.e., documents that have a very small maximum value in the topic-wise distribution. By removing them and re-running LDA, iVisCluster generally obtains significantly clearer topics. In PIVE, we performed this interaction near the 300th iteration (Fig. 9(b)), which is an early iteration when compared to the total number of iterations performed by LDA. However, such an interaction successfully generates clearer topics (Fig. 9(c)) over the standard approach where users have to wait for the algorithm to finish its full iterations in order to perform the same interaction.

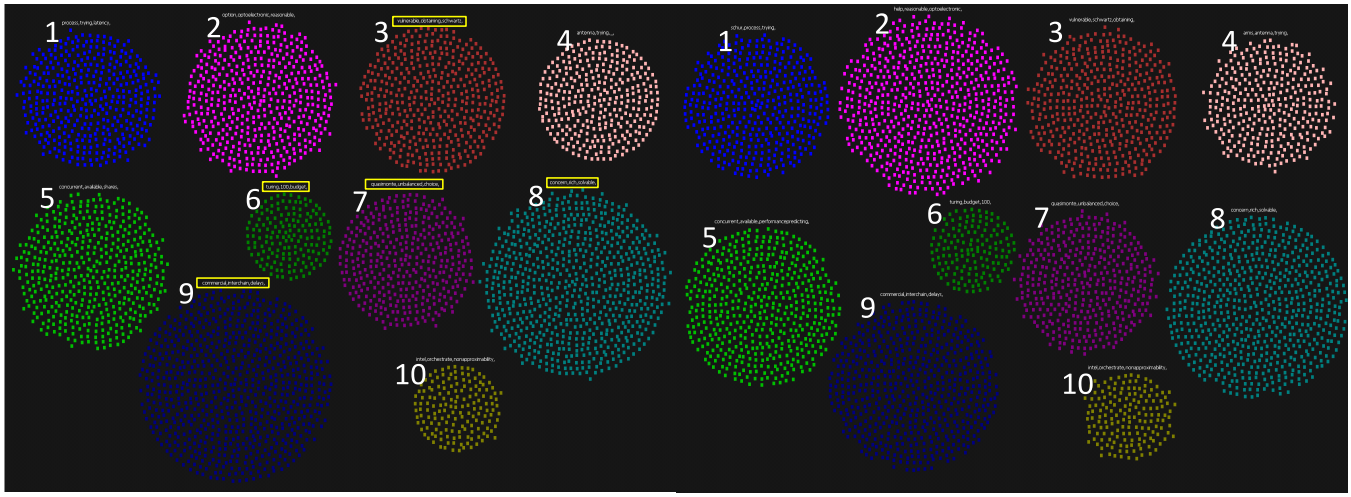
6 Conclusions and Future Work

We have presented PIVE (**P**er-**I**teration **V**isualization **E**nvironment for supporting real-time interactive visualization with computational methods). One of its apparent advantages is its ability to present users with the intermediate results during the interactions, which could reveal a significant amount of information immediately in visual analytics. Another important advantage is that it indeed opens up the possibility of performing small multiple interactions, which in the past have been considered to be too inefficient, and allows the real-time control over computational methods in visual analytics. In fact, the interactions we proposed in this paper are relatively simple, which do not involve any major algorithmic modifications, but after a sequence of interactions, the results reflects the intention of users sufficiently well in real-time. In this sense, PIVE makes them significantly useful by enabling users to perform these interactions easily and efficiently.



(a) The second-iteration result

(b) The sixth-iteration result



(c) The converged (25th-iteration) result after fixing clusters

(d) The converged (26th-iteration) result without any interaction

Fig. 7: The results of the PIVE integration of *k*-means in Jigsaw. At the sixth iteration, the interaction of fixing the yellow-colored clusters is made (b). The final result with and without this interaction is shown in (c) and (d), respectively. The NSF-awarded abstract data have been used. The detailed keyword summary is shown in Table 1

However, the advantage of our framework can be limited when the changes between iterations remain nontrivial, resulting in inconsistent visualizations. We have seen this kinds of limitations when using LDA under PIVE due to the random nature of the used LDA algorithm. As a future work, we plan to tackle this problem more actively by, for example, post-processing the results or even imposing additional constraints in computational methods so that the results from the following iterations do not change much from the current ones. Finally, another interesting research direction we will pursue is to extend PIVE to various parallelized computational algorithms for the large-scale data visual analytics.

REFERENCES

[1] J. Alsakran, Y. Chen, Y. Zhao, J. Yang, and D. Luo. Streamit: Dynamic visualization and interactive exploration of text streams. In *Pacific Visualization Symposium (PacificVis), 2011 IEEE*, pages 131–138, 2011.

[2] S. Basu, A. Banerjee, and R. J. Mooney. Semi-supervised clustering by seeding. In *Proceedings of the Nineteenth International Conference on Machine Learning*, ICML '02, pages 27–34, San Francisco, CA, USA, 2002. Morgan Kaufmann Publishers Inc.

[3] C. Bishop. *Pattern recognition and machine learning*. Springer, 2006.

[4] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, March 2003.

[5] A. Buja, D. Cook, and D. Swayne. Interactive high-dimensional data visualization. *Journal of Computational and Graphical Statistics*, 5(1):78–99, 1996.

[6] J. Cottam, A. Lumsdaine, and C. Weaver. Watch this: A taxonomy for dynamic data visualization. In *Visual Analytics Science and Technology (VAST), 2012 IEEE Conference on*, pages 193–202, 2012.

[7] T. F. Cox and M. A. A. Cox. *Multidimensional Scaling*. Chapman & Hall/CRC, London, 2000.

[8] J. De Leeuw. Applications of convex analysis to multidimensional scaling. *Recent developments in statistics*, pages 133–146, 1977.

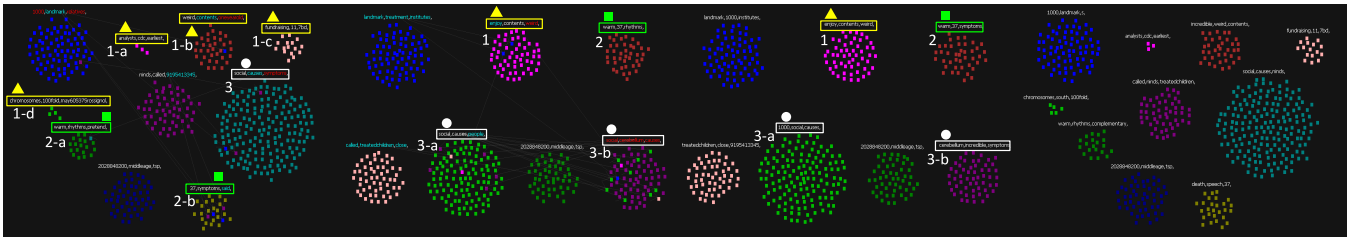
[9] W. Dou, X. Wang, R. Chang, and W. Ribarsky. Paralleltopics: A probabilistic approach to exploring document collections. In *Visual Analytics Science and Technology (VAST), 2011 IEEE Conference on*, pages 231–240, oct. 2011.

[10] G. Ellis and A. Dix. Enabling automatic clutter reduction in parallel coordinate plots. *Visualization and Computer Graphics, IEEE Transactions on*, 12(5):717–724, 2006.

[11] A. Endert, C. Han, D. Maiti, L. House, S. Leman, and C. North. Observation-level interaction with statistical models for visual analytics. In *Visual Analytics Science and Technology (VAST), 2011 IEEE Conference on*, pages 121–130, 2011.

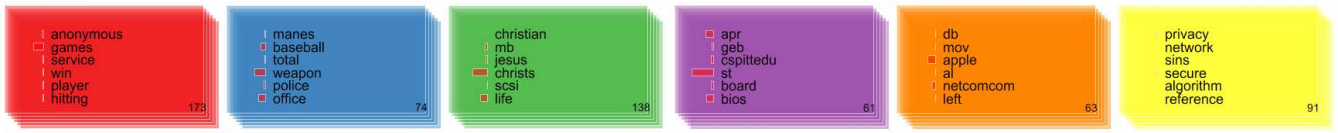
[12] G. Faconti and M. Massink. Continuous interaction with computers: Issues and requirements. *Vol. 3 of the proc. of HCI International 2001*, pages 301–305, 2001.

[13] D. Fisher, I. Popov, S. Drucker, and m. schraefel. Trust me, i'm partially

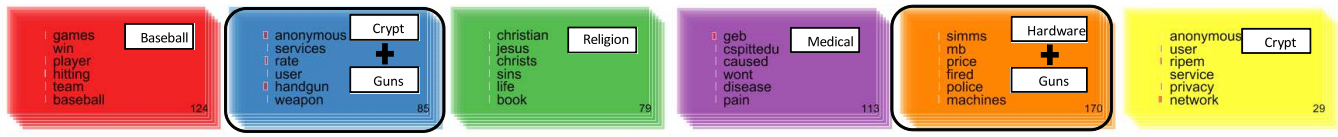


(a) The third-iteration result (b) The fourth-iteration result after split/merge interactions (c) The final (15th-iteration) result with split/merge interactions (d) The final (7th-iteration) result without split/merge interactions

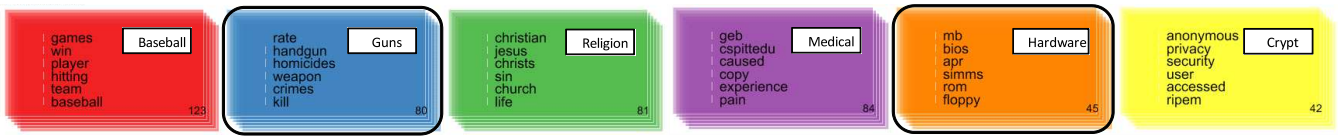
Fig. 8: An example of split/merge interactions. The yellow and green ones in (a) are merged to the same-colored ones, respectively, in (b), and the white one in (a) is split to the-same colored ones in (b). Webpages about autism have been used as an input data set. The detailed keyword summary is shown in Table 2



(a) The third-iteration result



(b) The 300th-iteration result



(c) The 700th-iteration result

Fig. 9: An example of filtering documents whose cluster memberships are unclear. This interaction is done in the 300th iteration, and the topics become clearer in the later iterations. 20 newsgroups data have been used.

right: incremental visualization lets analysts explore large datasets faster. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '12, pages 1673–1682, New York, NY, USA, 2012. ACM.

[14] G. H. Golub and C. F. van Loan. *Matrix Computations, third edition*. Johns Hopkins University Press, Baltimore, 1996.

[15] I. T. Jolliffe. *Principal Component Analysis*. Springer-Verlag, New York, 1986.

[16] I. T. Jolliffe. *Principal component analysis*. Springer, 2002.

[17] D. Keim. Information visualization and visual data mining. *Visualization and Computer Graphics, IEEE Transactions on*, 8(1):1 –8, jan/mar 2002.

[18] H. Lee, J. Kihm, J. Choo, J. Stasko, and H. Park. iVisClustering: An interactive visual document clustering via topic modeling. *Computer Graphics Forum*, 31(3pt3):1155–1164, 2012.

[19] K.-L. Ma. In situ visualization at extreme scale: Challenges and opportunities. *Computer Graphics and Applications, IEEE*, 29(6):14–19, 2009.

[20] A. K. McCallum. Mallet: A machine learning for language toolkit. http://mallet.cs.umass.edu, 2002.

[21] H. Piringer, C. Tominski, P. Muigg, and W. Berger. A multi-threading architecture to support interactive visual exploration. *Visualization and Computer Graphics, IEEE Transactions on*, 15(6):1113–1120, 2009.

[22] I. Porteous, D. Newman, A. Ihler, A. Asuncion, P. Smyth, and M. Welling. Fast collapsed gibbs sampling for latent dirichlet allocation. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '08, pages 569–577, New York, NY, USA, 2008. ACM.

[23] J. Seo and B. Shneiderman. Interactively exploring hierarchical clustering results [gene identification]. *Computer*, 35(7):80 –86, jul 2002.

[24] R. Spence and A. Press. Information visualization. 2000.

[25] J. Thomas and K. Cook. *Illuminating the path: The research and development agenda for visual analytics*, volume 54. IEEE, 2005.

[26] T. Tu, H. Yu, L. Ramirez-Guzman, J. Bielak, O. Ghattas, K.-L. Ma, and D. R. O'Hallaron. From mesh generation to scientific visualization: an end-to-end approach to parallel supercomputing. In *Proceedings of the 2006 ACM/IEEE conference on Supercomputing*, SC '06, New York, NY, USA, 2006. ACM.

[27] L. van der Maaten and G. Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605, 2008.

[28] F. Wei, S. Liu, Y. Song, S. Pan, M. X. Zhou, W. Qian, L. Shi, L. Tan, and Q. Zhang. Tiara: a visual exploratory text analytic system. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '10, pages 153–162, New York, NY, USA, 2010. ACM.

[29] M. Williams and T. Munzner. Steerable, progressive multidimensional scaling. In *Information Visualization, 2004. INFOVIS 2004. IEEE Symposium on*, pages 57 –64, 0-0 2004.

[30] P. C. Wong, H. Foote, D. Adams, W. Cowley, and J. Thomas. Dynamic visualization of transient data streams. In *Information Visualization, 2003. INFOVIS 2003. IEEE Symposium on*, pages 97–104, 2003.

[31] Z. Xie, M. Ward, and E. Rundensteiner. Visual exploration of stream pattern changes using a data-driven framework. In G. Bebis, R. Boyle, B. Parvin, D. Koracin, R. Chung, R. Hammound, M. Hussain, T. Kar-Han, R. Crawfis, D. Thalmann, D. Kao, and L. Avila, editors, *Advances in Visual Computing*, volume 6454 of *Lecture Notes in Computer Science*, pages 522–532. Springer Berlin Heidelberg, 2010.

[32] H. Yu, C. Wang, R. Grout, J. Chen, and K.-L. Ma. In situ visualization

for large-scale combustion simulations. *Computer Graphics and Applications, IEEE*, 30(3):45–57, 2010.