Int J Software Informatics, Volume 10, Issue 3 (2016), pp. 000–000
International Journal of Software and Informatics, ISSN 1673-7288

E-mail: ijsi@iscas.ac.cn
http://www.ijsi.org
Tel: +86-10-62661048

DOI: 10.21655/ijsi.1673-7288.00230

A Framework for Interactive t-SNE Clustering

Jared Bond, Christan Grant, Joshua Imbriani and Erik Holbrook

(University of Oklahoma, School of Computer Science Norman, Oklahoma)

Abstract In this paper, we describe our progress in creating the framework for an interactive application that allows humans to actively participate in a t-SNE clustering process. t-SNE (t-Distributed Stochastic Neighbor Embedding) is a dimensionality reduction technique that maps high dimensional data sets to lower dimensions that can then be visualized for human interpretation. By prompting users to monitor outlying points during the t-SNE clustering process, we hypothesize that users may be able to make clustering faster and more accurate than purely algorithmic methods. Further research would test these hypotheses directly. We would also attempt to decrease the lag time between the various components of our application and develop an intuitive approach for humans to aid in clustering unlabeled data. Research into human assisted clustering can combine the strengths of both humans and computer programs to improve the results of data analysis.
Key words: t-SNE; clustering; interactive analytics

Bond J, Grant C, Imbriani J, Holbrook E. A framework for interactive t-SNE clustering. *Int J Software Informatics*, **Vol.10, No.3 (2016): 000–000. http://www.ijsi. org/1673-7288/10/230.htm**

1 Introduction

Computers have surpassed humans in the completion of several complex tasks[1], leaving humans out of most algorithm processing loops. By the time humans can perceive errors, process corrections, and submit fixes, most iterative algorithms will have already moved forward. When the amount of data to be processed is large or rapidly changing, accurately visualizing high dimensional data sets for human interpretation presents a challenging problem[3,7,9].

Many researchers have sought to map high dimensional data to lower dimensional representations that humans can perceive[2,4,8]. These techniques attempt to preserve the patterns in as much of the original data's structure as possible. While the academic community has focused on algorithmic approaches to visualizing data, there has been relatively less research into providing human users with interactive roles in algorithmic processes[5].

Clustering, often one of the first methods employed for exploratory data analysis, algorithmically classifies data into logical groups. Though many improvements have been made to clustering processes, it remains difficult to produce quality clusters.

Corresponding author: Christan Grant, University of Oklahoma, School of Computer Science, 110 W. Boyd, DEH 234, Norman, Oklahoma, 73019. Email: cgrant@ou.edu
Received 2016-08-15; Revised 2016-10-01; Accepted 2016-10-28.

Data scientists often rely on the output of a clustering algorithm over many iterations to properly develop clusters. This creates a human-algorithm workflow for analyzing data that falls short of fully engaging the human. Instead of only interacting with the algorithm by adjusting parameters, humans should be able to actively participate in the clustering process alongside the algorithm.

In this paper, we discuss our current progress in creating an interactive application that allows users to aid in a t-SNE clustering process (see Figure 1). t-SNE (t-Distributed Stochastic Neighbor Embedding) is an efficient dimensionality reduction algorithm that enables humans to interpret low dimensional transformations of high dimensional data sets[11]. In particular, we establish the ability for users to monitor outliers, since these are the points that the algorithm is likely struggling to classify. We believe that enabling humans to aid in the clustering process holds promise for clustering data both more accurately and more quickly than existing methods that rely solely on algorithmic methods.

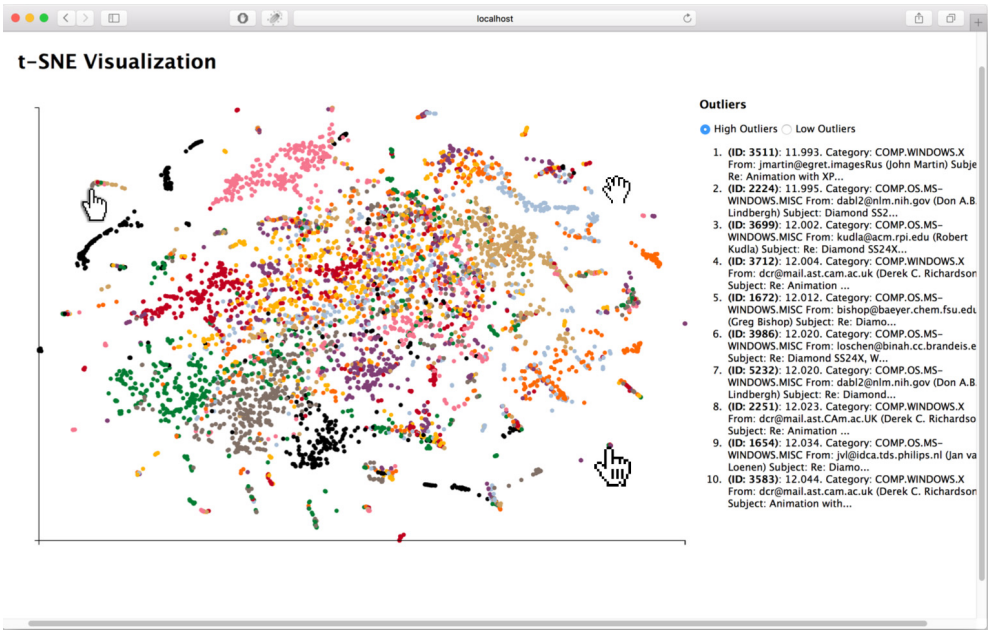

Figure 1. Interactive t-SNE Visualization. Our application supports multiple users working together to assist t-SNE in clustering data, represented by the various cursors. Hovering over a point brings up a tooltip that displays the point's relevant text. The points are colored to correspond to the categories in the 20 Newsgroups dataset. The right panel allows users to toggle between viewing high and low outliers.

2 Background

t-SNE provides a method for embedding a high dimensional data set X into a 2- or 3-D representation Y that can be plotted[11]. Unlike previous linear data reduction techniques that rely on forcing dissimilar points to be far apart (e.g. PCA), t-SNE relies on keeping similar points close together. Ensuring that similar points remain close together results in better visualizations of nonlinear data sets by preserving local

structure. Because many modern data sets are nonlinear, t-SNE thus provides a more reliable method for clustering than pre-existing methods.

To quantify the notion of similarity, t-SNE begins by computing the conditional probability that each point $x_i \in X$ in the high dimensional data set would choose its neighbor $x_j \in X$ under a Gaussian distribution centered on x_i. This conditional probability is given by:

$$p_{j|i} = \frac{\exp(-||x_i - x_j||^2/2\sigma_i^2)}{\sum_{k \neq i} \exp(-||x_i - x_k||^2/2\sigma_i^2)}$$

Points that are close together in space therefore produce high similarity probabilities while points that are far apart have correspondingly low probabilities. We then compute the joint probabilities p_{ij} in the high dimensional space as symmetrized conditional probabilities, $p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n}$.

We proceed to generate the probabilities in the low dimensional space according to a Student t-distribution, with each point q_{ij} given by:

$$q_{ij} = \frac{(1 + ||y_i - y_j||^2)^{-1}}{\sum_{k \neq l}(1 + ||y_k - y_l||^2)^{-1}}$$

Due to its fat tails, the Student t-distribution allows for distances to be amplified to prevent crowding in the center of the plot, and it carries the additional advantage of being faster to compute than a Gaussian distribution since no exponentials are involved.

The final step in the t-SNE process is to minimize the differences between the high dimensional probabilities and the low dimensional probabilities. This is accomplished by performing gradient descent on the Kullbach-Leibler Divergence, which can be thought of as a ratio between the high and low dimensional joint probabilities that preserves local structure by heavily penalizing similar points that are incorrectly modeled as being far apart from each other. The gradient is given by:

$$\frac{\delta C}{\delta y_i} = 4 \sum_j (p_{ij} - q_{ij})(1 + ||y_i - y_j||^2)^{-1}(y_i - y_j)$$

Because the Kullbach-Leibler Divergence is computationally expensive (running in $O(n^2)$ time), the exact version of t-SNE is infeasible for even modestly large data sets. To improve upon the original t-SNE algorithm, one can use a Barnes-Hut approximation that reduces the time complexity substantially to $O(n \log n)$[10]. This approximation uses a quadtree data structure where each node stores the center of mass of its children. This summary value provides a good approximation of the true joint probabilities when the data points are relatively close together, resulting in the significant improvement in time complexity. Barnes-Hut t-SNE therefore provides a feasible method for visualizing data sets containing millions of points.

In this paper we describe a system for users to interact with t-SNE as it clusters data. With large data sets, the clustering process can still take a substantial amount of time to terminate, and the algorithm can struggle with clustering boundary points that humans may be able to identify more easily.

3 Architecture

Our system accomplishes two goals: it displays an animated plot of points as they are clustered by t-SNE, and it allows a user to move one or more points in the client and incorporate the changes into the running t-SNE algorithm. As illustrated in Fig. 2, our system is organized into four main components: a real-time database, a modified t-SNE algorithm, a non-blocking I/O server that handles communication between the client and database, and a web client.

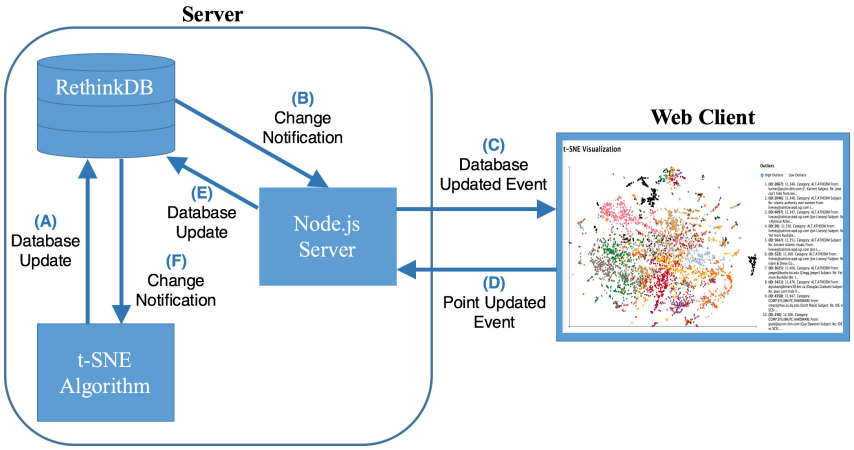

Figure 2. Interactive t-SNE Visualization Architecture. The t-SNE algorithm runs in a Python script on the server, committing intermediate results to the database (A). The database saves these changes, firing a change notification (B) to the Node server which then alerts the client of the update (C) so that it can animate the points' movement. Similarly, users can update points in the client (D), which prompts the listening Node server to update the database (E). When this happens, the database fires a change notification (F) that is detected by a dedicated thread in the Python script which updates the local values of the points for the t-SNE process.

We use the open-source database RethinkDB to store data and perform real-time queries. RethinkDB features changefeeds, which automatically alert listening programs when the database changes and pass along the updated values. This feature removes the need to poll the database for changes, making it much easier to write interactive client-server applications.

Our t-SNE algorithm is forked from scikit-learn's Python implementation[6]. We modify their t-SNE script in order to commit the current (x, y) coordinates of each point to our database after every i^{th} iteration through the algorithm's gradient descent. This enables the client to display a constantly updated plot of the points as they are clustered by the t-SNE algorithm. Additionally, we create a separate thread in the t-SNE script that listens for client-side updates in the database. In this way, we can update the (x, y) coordinates of the points in the algorithm to reflect any changes that the user (or users) makes on the client.

We also implement a Node.js server to monitor changes in both the database and the client. We use Node.js for our server because it is readily compatible with both our client and our database and because it handles I/O asynchronously, allowing it to

support multiple clients well. When the algorithm updates the database with adjusted (x, y) coordinates, the server passes the new values to the client. Similarly, when a user updates the location of a point on the client, the server commits the changes to the database, keeping the two in sync. Finally, we animate the data points in a browser application in response to the running t-SNE algorithm. The data is modeled using D3.js, a JavaScript library that provides extensive support for manipulating and animating data.

While we have only used our application with single users, our architecture is scalable to support multiple clients monitoring the same data set. Just as single users could aid in clustering, multiple users could cooperate to distribute the workload among themselves and further speed the process. To allow users to take an active role in the clustering algorithm, we direct them to track the top and bottom k outliers, given by the points that have moved the most and least over the course of the clustering process. Using AngularJS's two-way data-binding features, we track the top k outliers since it is likely that they are composed of oscillating points, or data that the algorithm is struggling to cluster by itself. Similarly, the bottom k outliers are points that could be surrounded by dissimilar points, preventing them from moving, or they could be points that the algorithm leaves alone when it is unable to cluster them confidently.

4 Experiments

In our experiments, we measured the performance differences between our system with no interaction and a purely algorithmic approach in order to gauge our system's baseline performance. First, we measured the runtime of scikit-learn's unmodified t-SNE script to serve as a control group. Then, we measured the runtime of our modified t-SNE script with visualization and no human interaction. Finally, we measured the lag time from when points are committed by t-SNE to the database and when the client registers the changes.

We used various subsets of the 20 Newsgroups data set, which consists of 11,314 news items labeled according to one of twenty categories. We used labeled data so that we could visually gauge the t-SNE algorithm's ability to accurately cluster data in addition to measuring the various runtimes. The following experiments were run on the data set after first using Truncated SVD to reduce the dimensionality to fifty in order to speed up computation times and smooth out noise in the data. We used the default values for all of the parameters except perplexity, which we set to 40 to reflect the density of our data set. All experiments were run on a 2015 MacBook Pro with a 2.5GHz quad-core i7 processor.

Figure 3 displays the runtimes of the unmodified t-SNE algorithm and the modified t-SNE algorithm with visualization. The x-axis plots the total number of categories used from the 20 Newsgroups data set, i.e., x = 4 corresponds to a four category subset of the total data set. The y-axis plots the time in seconds for the algorithm to terminate. Our data indicate that the comparative trustworthiness of the two t-SNE algorithms (where values closer to 1 indicate greater fidelity to the original data set) is essentially the same, confirming that the underlying clustering process is identical for both versions.

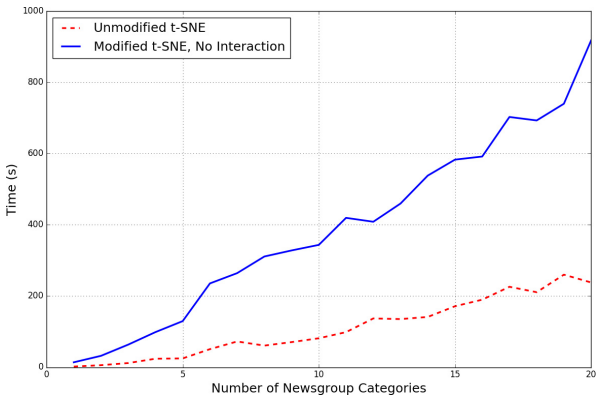

Figure 3. t-SNE Runtimes. Compared to the unmodified t-SNE algorithm implemented in Scikit-Learn, our modified t-SNE algorithm without human interaction is slower due to the cost of committing intermediate changes to the database for the client to animate. However, by visualizing changes and enabling human involvement, this framework could be valuable if human participation speeds clustering, aids in cluster accuracy, or provides insight into the underlying structure of the data.

Figure 4 displays the time for the database to process updates, along with the total event processing time (given by actions $(A) + (B) + (C)$ in Fig. 2). Again, the x-axis plots the total number of categories used for that simulation, where each set is a subset of the total 20 Newsgroups data set. The y-axis plots time in milliseconds. While the total end-to-end time exhibits a dip at the upper category levels relative to the database processing time, this is most likely attributable to variations in connectivity and the amount of tasks running in the background. Alternatively, this dip could indicate that with larger data sets, the computational power necessary for clustering requires the MacBook Pro to devote more processing cores to the task than with smaller data sets, possibly indicating that better performance could be achieved through parallelization.

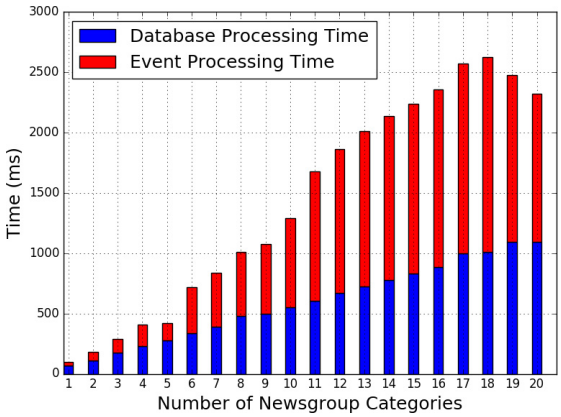

Figure 4. Lag Between Server and Client. This graph charts the database lag time along with the time to process events, given by $(A) + (B) + (C)$ in Figure 2. While the database incurs some time to process updates, this figure shows that event communication costs become the main bottleneck as the size of the dataset grows.

Finally, our initial, qualitative findings are that the browser-based client provides an intuitive and useful mechanism for enabling human users to become active participants in the clustering process. Humans are able to drag either single points or multiple points to the cluster in which they belong, and the algorithm is able to incorporate these updated coordinates while continuing its gradient descent optimization.

5 Discussion

While our visualization of the t-SNE algorithm slowed the computation time, we believe that visualizing the process provides benefits that make the slowdowns a worthwhile tradeoff compared to simply running the algorithm from the console. Additionally, we believe that much of this slowdown is due to both the server and client running on a local machine instead of using a dedicated remote server, so it is likely that our application's performance could be improved. In visualizing the process, users can interact with the algorithm by monitoring outlying points and tracking the movement of the data over time. The visual shapes of clusters and the identification of problem points could potentially provide useful insights into the underlying structure of the data that a non-interactive, non-visual approach would lack.

Our visualized approach still has much room for improvement, however. In particular, the lag time between the t-SNE algorithm and the client should be improved in order to make the application more responsive to user input. While we were able to use the linear regression equation generated for the end-to-end lag time to add a delay function to the data points' animation speeds to smooth over data discrepancies, more research should be done to correct the heart of the problem. Additionally, more research should be done to handle unlabeled data. While we are confident that our approach can easily identify outliers for users to monitor, unlabeled points would be harder for users to classify than labeled points. Visually describing summary information for each forming cluster is thus a desirable next step to improve users' ability to aid the algorithm in clustering unlabeled data. Once these changes to the system are finalized, we can then begin experiments with human users to directly compare usability and accuracy between the purely algorithmic method and the interactive version.

6 Summary

In this paper, we created an interactive t-SNE visualization application in order to provide human users with meaningful roles in clustering processes. While our visualization platform is slower than the algorithm running by itself, it provides the added benefits of animating the clustering process and involving users as active participants. Our initial assessment is that the application holds promise for continued research into involving humans in clustering.

Future work will focus on designing and performing user acceptance tests and modifying the interface in response to user feedback. We are particularly concerned with obtaining quantitative data to determine if human users can make a significant impact on the speed and accuracy of the clustering process. This research should

clarify the role that humans can serve in clustering by testing the efficacy of various human actions: confirming existing clusters, creating new cluster centers, or scattering existing clusters that have been classified incorrectly. Additionally, more work must be done to improve the lag times of our application in order to make the process more responsive to user input. Managing the number of points on the screen at once could address this issue by collapsing nearby points together when the view is zoomed out and separating them apart again when the view is zoomed in. We may be able to leverage the underlying quadtree data structure in the Barnes-Hut algorithm to manage these groupings. This could also alleviate the cognitive load users experience when clustering a large number of points. Finally, future work should explore the best methods for involving human users in aiding t-SNE when clustering unlabeled data, including representing data with different glyphs instead of points.

Keeping human users involved in data analysis has the potential to make clustering both faster and more accurate than current methods that are solely algorithmic. Our work seeks to maintain a place for humans in data analysis rather than ceding all analytical work to software programs. By combining the capabilities of both humans and computers, our understanding of data and its applications for society can only increase.

References

[1] Andersen E. Opening statement: Will computers out-compete us all?: The technological singularity (ubiquity symposium). Ubiquity, 2014 (October): 1:1–1:8. ISSN 1530–2180. 10.1145/2668424. http://doi.acm.org/10.1145/2668424.

[2] Borg I, Groenen PJF. Modern multidimensional scaling: Theory and applications. Springer Science & Business Media, 2005.

[3] Levy DM. To grow in wisdom: vannevar bush, information overload, and the life of leisure. Proc. of the 5th ACM/IEEE-CS Joint Conference on Digital Libraries. ACM. 2005. 281–286.

[4] Ng AY, Jordan MI, Weiss Y, et al. On spectral clustering: Analysis and an algorithm. Advances in Neural Information Processing Systems, 2002, 2: 849–856.

[5] Paul CL, Argenta C, Elm W, Endert A. Future directions of humans in big data research: Summary of the 1st workshop on human-centered big data research. 2014 IEEE International Conference on Big Data (Big Data). 2014.

[6] Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, Vanderplas J, Passos A, Cournapeau D, Brucher M, Perrot M, Duchesnay É. Scikit-learn: Machine learning in python. J. Mach. Learn. Res., November 2011, 12: 2825–2830. ISSN 1532–4435. http://dl.acm.org/citation.cfm?id=1953048.2078195.

[7] Pirolli P, Card S. The sensemaking process and leverage points for analyst technology as identified through cognitive task analysis. Proc. of International Conference on Intelligence Analysis, 2005, 5, 2–4.

[8] Tenenbaum JB, Silva VD, Langford JC. A global geometric framework for nonlinear dimensionality reduction. Science, 2000, 290(5500): 2319–2323.

[9] In: Thomas JJ, Cook KA, eds. Illuminating the Path: The Research and Development Agenda for Visual Analytics. IEEE Computer Society, 2005.

[10] van der Maaten L. Accelerating t-sne using tree-based algorithms. Journal of Machine Learning Research, 2014, 15: 3221–3245.

[11] van der Maaten L, Hinton G. Visualizing high-dimensional data using t-sne. Journal of Machine Learning Research, 2008, 9: 2579–2605.