Apply T-SNE Technique for Visual Analytics Tasks

Abdulrahman Alshammari

University of Delaware

***Abstract*— This report shows how we use t-SNE technique in many visual tasks in order to extract meaningful information. In order to achieve the main goal of this project, we used external libraries such as MPLD3 and Plotly.**

I. INTRODUCTION

T-Distributed Stochastic Neighbor Embedding (t-SNE) is a technique that is used for reducing the dimensions of a dataset and visualize the data in a convenience way. In this project, we apply clustering algorithms on the data and generate many sub-clusters. In each sub-cluster, we would like to highlight where the instances located during visualization. To do that, we apply t-SNE on each sub-cluster and try visualize only the data belong to the selected one as one color, and the rest of other sub-clusters as a different color. We will do the same process among all other sub-clusters. To run that, the t-SNE script is written in python that takes two parameters, the whole clustered data and the labels (sub-cluster). The expected result will be two dimensional plot with many points where represent instances from clustered data.

II. OBJECTIVES

In this project, we divide the main task of this project into sub tasks. These are:

- Generate a single image for each cluster to show where the points which belong to the selected cluster are distributed and how far they are from other clusters points.
- Generate a PDF file that contains all generated images.
- Selection the two colors that are used generated images is based on user preferences.
- T-SNE algorithm has certain parameters that affect on the overall result such as the maximum number of iterations and perplexity, we also make them up to the user preferences.
- Ability to Visualize the t-SNE result in a browser, so the user can interact with the result and get hidden information from exploring the result. We propose the following two main features:
  - Linked Brushing style provided by MPLD3.
  - Interactive legends provided by Plotly.
- All required scripts are written in Python.

III. REQUIREMENTS

There are many libraries that are required to view results in browser. Beside the basic libraries to execute the python scripts, the dataset should be in an appropriate design. Overall, these are the main requirements:

A. MPLD3

This library combines Matplotlib, the popular Python-based graphing library, and D3js, the JavaScript library for creating interactive data visualizations for the web. The expected result of this library is to view the matplotlib as HTML file. The main dependencies are jinja2 and Matplotlib. There are many different build-in plugins provided by MPLD3. For this project, we only use Linked Brushing.

B. Plotly

The purpose of using this library is to make the scatter plot that generated from t-SNE more interactive with the user. Plotly is a python graphical library that enables interactive visualization of a plot using a browser. In this project, we use only offline mode because it does not require having an account. Also, we apply interactive legend plugin to view t-SNE and be able to hide or show certain clusters points.

C. Dataset

In term of datasets, the t-SNE script requires the dataset to have headers, the attributes names. Also, the labels data should contain the sample name followed by its cluster number without headers in both attributes. In our dataset, the sample name is a malware name and cluster number starts from 1 to N, where N is the total of clusters and each number represents one sub-cluster.

IV. EXPERIMENT

For simplicity, we divide the outcome of this experiment into three main parts

A. Generate Images and a PDF file

One of the feature of t-SNE is that the results have different patterns in term of plot point distributions after each time we run t-SNE code. However, it is recommended to have one pattern if we plan to have comparison task among clusters. T-SNE experts have many different ways in order to achieve a single patter as a result. Some of them prefer to disabled random state option to lead a single pattern

after each time we run the code. Others prefer to have only one time run and generate any number of images from it by applying filtering techniques. The last option is more convenient for many reasons. First, we can still get many different patterns which is a feature of t-SNE, but we still have the same pattern of all images once we run the code one time. Second, there is a significant reduction of time when we run the code since we just need to run the code one time and we can generate images as many as we need from t-SNE result.

We modified the t-SNE code in a way that we only run the code once and apply filtering. Each image will have two colors on the plot points which corresponding whether the point belongs to a selected cluster or not. We do the process recursively over all clusters. As a result, we will combine all images in one pdf file for any further tasks. Also, the two colors of the plot points are given from the user who run the code as inputs even though we highly recommend to have red and lightgray as main colors. Any color defined by matplotlib is acceptable.

Figure 1 is one sample of the images we generate on provided datasets. The title shows that the image belongs to cluster number 1. The red plot points belong to cluster number 1 and gray represents all others clusters points.

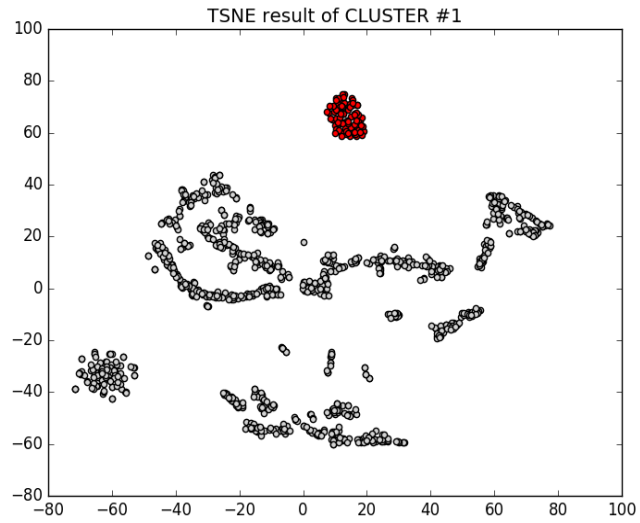

Fig. 1. One sample of the t-SNE images

B. View t-SNE using MPLD3

Beside generating images for each cluster, the second achieved task is to view the matplotlib scatter on a browser and let the user explore the hidden details of how cluster points close to each other. The applied technique is called Linked Brushing where we have n*n subplots. The diagonal subplots are displayed as a line that contains all cluster points in random way. The idea is to visualize a specific plot point on many different subplots in case it is hard to detect the point in one subplot. The way this style work is to make a square selection, then all remaining points outside the selection will be in one color (gray) in all subplots. Figure 2 shows the final output of using this plugin on t-SNE result.

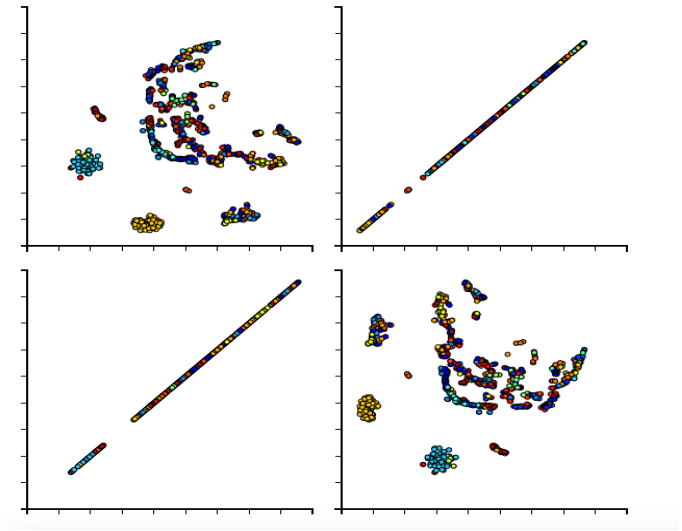

Fig. 2. The output of t-SNE using Linked Brushing provided by MPLD3

C. View t-SNE using Plotly

The second way to view the t-SNE result in a browser is to apply interactive legends to give control to user to determine which labels we want to view and which we do not. In specific, the result will contain interactive legends that contain a list of labels, each label represents one cluster. By default, all labels are selected. The user can hide a specific cluster by uncheck/deselect its label. Figure 3 show the default output of this Plotly feature. The Plotly result actually has three main parts.

1) The plot

It contains all points of selected clusters. Each cluster point is represented by one color. Coloring clusters are randomly selected so when we run the code in the second time, we expect to have different colors. However, all colors are distinguish and no two labels have the same color in the same result. When the user hover on any points, it will show to which labels a selected point belongs. The size of each point can be changed, but it is recommended to have points at the default size which is 6. Y-axis and X-axis are generated automatically based on how the points distributed.

2) Interactive legends

The main point of this legends is to make it interactive with user. If there are too many labels, the legend size is adjusted with the plot height and any extra labels can be reached using the scrollbar. In Figure 3, we can see only 22 labels where in fact there are 66 labels. There are two main task in this legend:

- Single click on a label will make the selected label visible or non-visible.

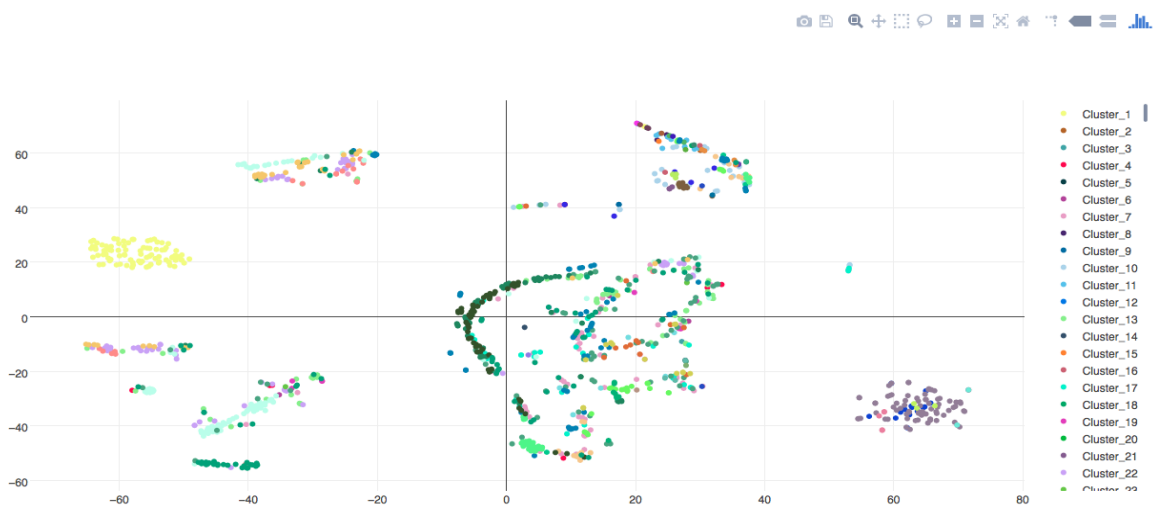

Fig. 3. The output of t-SNE using the interactive legends provided by Plotly

- Double click on a label will make all remaining labels visible or non-visible

3) *The Top-Right Menu*

The menu has some useful features that we can apply on the plot result. Most important features are:

- The user can hover at any vertical points and the plot will show any point that is located on the same x-axis to the hover point. The hover point is not required to be a real point; it might be a blank space.
- The user also can hover any point to get its label. Also, the user can get the x and y axises values.
- After the user zoom in or zoom out, the plot can be reset into its default zoom in one click.
- The user can also select certain points , and the plot will automatically hide any points that does not belong to the selection area
- The user can save the plot as an image with the latest update on plot.

V. LIMITATIONS AND CHALLENGSE

There are some issues that limits our work which are summarized as follow:

- The PDF file is extremely big in its size. Even thought each images are between 64 to 100 kb, the pdf size exceeds 200 mb.
- Some images seem to show only one color because the overlapping points. Even though some clusters have only one point, it should appear as at the top of others.
- Generating t-SNE images takes around 20 minutes based on the data we have.
- With different browsing tools for matplotlib, I was not able to combine MPLD3 and Plotly result in one HTML file.
- Plotly offline is free but it is limited if we compared with the other features provided by premium accounts in Plotly.
- We still learn Python on which the project based.

VI. FUTURE WORK

There are some significant points might be helpful if they are added in future.

- Make GUI or HTML that takes the dataset, the labels, and other t-SNE parameters from user and get all result in one browser. At this level of work, the tool is based on line commands from terminal and all related files and data have to be in the same folder.
- It will be helpful if the tool is able to change the color without run the tool again because the color preference might differ from one user to another.
- Design photo gallery to gather all images from t-SNE result.
- Improve the time and space complexity of the code.

REFERENCES

[1] Laurens van der Maaten, Geoffrey Hinton, Visualizing Data using t-SNE, `http://www.jmlr.org/papers/volume9/vandermaaten08a/vandermaaten08a.pdf`, vol. 9, 2008, Journal of Machine Learning Research.
[2] Laurens van der Maaten, t-SNE Laurens van der Maaten. `https://lvdmaaten.github.io/tsne/`, 2017.
[3] mpld3 developers, Bringing Matplotlib to the Browser `https://mpld3.github.io`, 2014.
[4] Plotly developers, Plotly Python Library `https://plot.ly/python/`, 2015.