Dahlia: Instant Visualization Library of Twitter Data in Python

Yitong Wang YW652@NYU.EDU
Center for Data Science, 726 Broadway, New York, NY 10003 USA

Meihao Chen MC5283@NYU.EDU
Center for Data Science, 726 Broadway, New York, NY 10003 USA

Abstract

Dahlia is designed to automate instant visualization of Twitter Data in Python. As an open, public online space, Twitter is occupied with the latest news and trendy topics. More and more people and professionals use Twitter as an approach to target their audiences and learn major popular news. Dealing with raw ”big data” can be incredibly annoying, and we believe that we can automate instant visualization to make life easier. We hereby introduce a visualization library in Python to generate visualization of Twitter data in a variety of formats, so that the information contained is presented clearly and directly.

1. Introduction

Dahlia is a library, i.e. a collection of functions that users import into Python programs. Using Dahlia therefore requires very fundamental Python programming knowledge, and basic knowledge in command-line interface operation. However, we do understand that Dahlia is designed for lots of professionals with only fundamental knowledge of programming, and therefore we try to scale its dependency on Python to a minimum.

The Python programming language is establishing itself as one of the most popular languages for scientific computing and general use. Besides its high-level interactive nature and its maturing ecosystem of scientific libraries, it is also an appealing choice for exploratory data analysis (Dubois, 2007),(Milmann & Avaizis, 2011) and down-to-earth data visualization. Yet, as a general-purpose language, it is increasingly used not only in academic settings but also in industry.

Proceedings of the 32nd International Conference on Machine Learning, Lille, France, 2015. JMLR: W&CP volume 37. Copyright 2015 by the author(s).

Dahlia harnesses a rich source of existing visualization techniques to provide a diverse range of visualization that reflects the underlying information in a variety of format, while maintaining an easy-to-use interface tightly integrated with the Python language. This answers the growing need for instant, professional visualization by non-specialists in the data science and analysis community, as well as in fields outside data science, such as marketing and business. Dahlia is different from other existing visualization/graphing libraries in Python for the following reasons:

- No pre-processing the data before use
- Capability to host large dataset
- Automated process, very easy to implement
- Interactive plots, with friendly user interface

We are going to detailing out the library functions more specifically in later sections.

2. Related Work

There are many libraries in Python built as visualization tools for scientific computing and data analysis, such as matplotlib (Hunter, 2007), Bokeh [1] and Plotly [2] . They are efficient and easy to get your hands on, yet they do not offer much room for interactions and are unable to handle large dataset.These libraries are more suitable simple, direct visualization of scientific computation.

CartoDB[3], the commonly adopted geographical visualization tool has recently launched CartoDB Twitter Maps to perform geospatial analysis and visualization of Twitter activity. With this new feature users can search for the Twitter activity they would like to visualize and analyze,

[1]https://github.com/bokeh/bokeh
[2]https://github.com/plotly/plotly.py
[3]https://cartodb.com/

select the period of tweets they want, and start mapping almost instantly. Dahlia functions differently since Dahlia emphasizes the importance of understanding dataset in a descriptive way, while CartoDB focuses specially on geotagged data.

On the other hand, there have also been many parallel work dedicated to interactive visualization of data, such as d3 (M.Bostock et al., 2011) and Tableau[4]. One of the major problems with these libraries is that they require much work into data processing and knowledge in JavaScript programming (especially in d3), plus limitations in dataset size, which creates even more trouble than the task itself for the non-technical professionals . Dahlia therefore becomes the perfect alternative by handling all above issues. In addition , we also add checkbox functions to some of our graphs for visualizations of more fields in the dataset. Checkbox allows user to select the data/field of his/her own choice, and by plotting different data in the same format enables more direct and insightful information extraction.

3. Underlying Technology

All graphing functions were written in Javascript, and we built them based on the existing functions in d3 (M.Bostock et al., 2011) and nvd3 [5], with much modifications and additions of necessary features to the original codes.The final product is presented in the form of a online dashboard, with selections of different data of user's choice, interactive checkboxes and mouse-over/mouse-out functionality for better user experience.

3.1. Data Acquisition

All data should be fetched from official Twitter API in order to maintain the standard format of data. Complete dataset should be compressed to the form of .rar or .json.zip in order to be passed smoothly into main functions of Dahlia. Dahlia does not contain functions or capability to fetch data for user, which means that user should have data ready before using Dahlia for further visualization.

3.2. Data Pipeline

Each graph takes input in different format, and we have to customize the data processing for each graph. As this is a visualization tool, we currently do not account to data acquisition, and all statements and methods from here now assume that the data is provided.

Since we are visualizing the twitter data, and therefore all data passed in should be raw data extracted from Twitter API, without any pre-processing (as this will disturb and affect the built-in functions of the library). Ideally, the main function takes the path of the raw data, which is compressed in form of .rar or .json.zip. We grasp all fields needed for subsequent visualization functions.

What should be noted here is that we decide not to consult any external techniques or functions that falls outside Python capabilities, such as MapReduce or jq (json fields extraction tools). The running commands or installation of these external tools can be fairly confusing for users, let alone the fact that using them will generate an abundance of temporary files, which take up lots of unnecessary memory space. All data processed from the pipeline will be named and put under a pre-designated directory, from where they will be later called by the drawing function .

3.3. Function Designs

All graphing functions are written in JavaScript, and we utilize libraries such as D3, nvd3, jQuery and NodeJS. Each function takes data processed from pipeline. As JavaScript is limited in hosting capacity and data processing efficiency, all data passed in has been reduced to its minimum in terms of structures for the purpose of "instant" visualization.

3.4. Hosting Environment

Instead of having the graph showing up on the screen, we put all graphs collectively on a webpage, as we believe that web hosting is more suitable for interactive graphs than local, and users have more control towards their preferences of how information is presented. The online dashboard looks like the following: Seen from Figure 1, The web

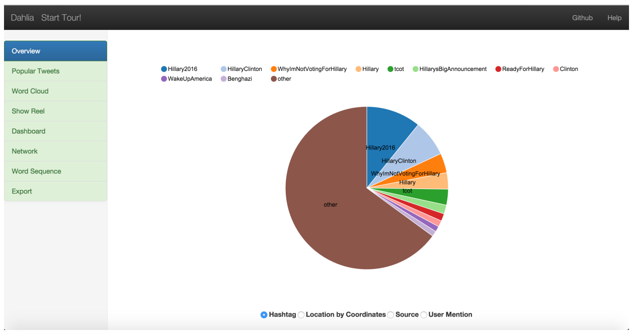


Figure 1. Overview of the online dashboard

hosting allows a variety of external handy functions, including a thorough tour of the dashboard, the link to Github repository and detailed wiki of Dahlia. On the left is a drop-

[4]http://www.tableau.com/
[5]https://github.com/novus/nvd3

down list of all graphs presented, and user can click on any graph of their choices.

4. Example Illustrations

As Dahlia only provides a limited number of graphs compared to D3 and Tableau, we will be presenting and illustrating each graph more in-depth in this section. The sample data employed are tweets regarding Hillary Clinton in the first 48 hours of her presidential announcement in Apr 2015. The entire dataset is compressed in .rar format and of total size of 1GB. The loading and processing of this dataset takes about 8-10 min, which gives a ballpark of processing efficiency.

4.1. Interactive Pie Chart

Interactive Pie Chart is inspired by the pie chart created in nvd3. We provide 4 source datasets for visualization: hashtag, source, user mentions and geographical locations, which are self-explanatory by names. This interactive piechart lists out the top 10 values of the selected field and all legends are color-coded. Mouse-over to the pie chart will accentuate the selected portion and displays the exact count of the slice. Moreover, the user can click ”off” any pie slice by clicking on its legend and the graph will be updated using the updated data. The 2 graphs below display the in-place update of pie chart when clicking off one legend. Each slice is adjusted accordingly when ”other” (brown portion in Figure 2) is clicked off. This handy functions comes in helpful when there are disproportionate noises and biases in the dataset.

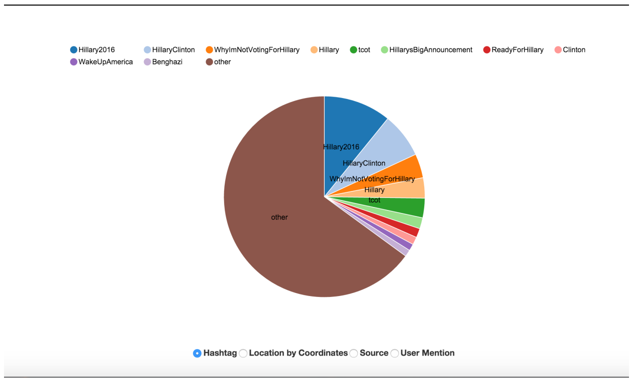


Figure 2. Before clicking-off legend

4.2. Stacked Area Chart

Stacked Area Chart consists of 3 types of area charts: stacked, stream and expanded. It graphs time series change in counts and percentages for hashtag, user mentions and source. Exact values of fields will be displayed when mouse-over to any part of the graph, and user can click on legend or either portion of the graph to visualize that portion exclusively.

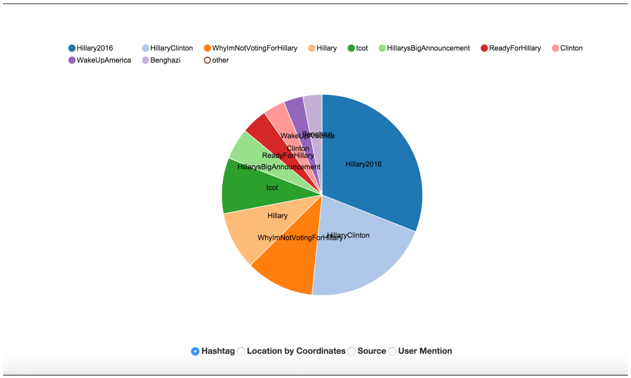


Figure 3. After clicking-off legend

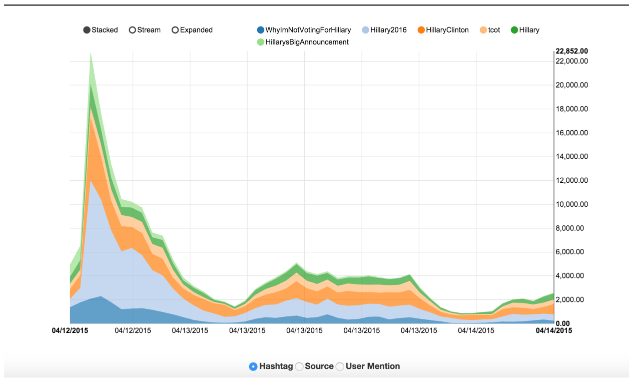


Figure 4. Stacked Graph

Stacked area chart gives a comprehensive comparison between counts of each item of the selected field. Stream graph, in addition to the previous one, displaces all items around a central axis, resulting in a more flowing, organic shape. Expanded graph utilizes the whole graphing area as an entity and graphs the relative percentage of each item, which displays the more granular comparisons between items.

4.3. Popular Tweets

This graph displays the top N most popular tweets in the dataset. On top of the graph is the control slider that adjusts the number of tweets displayed. Each tweet is accompanied by its corresponding count in the dataset. From this graph can the user tell the most popular tweet in the twitter space and therefore infer the most popular topics that are

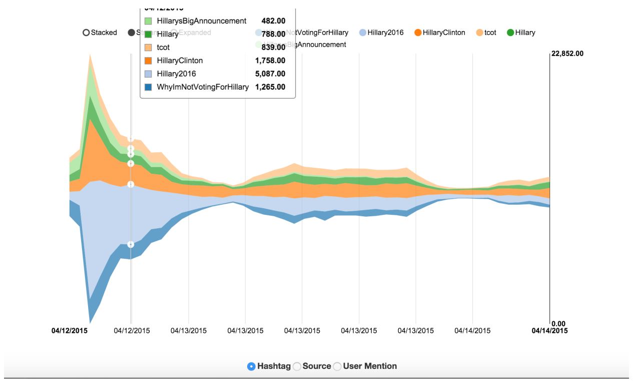


Figure 5. Stream Graph

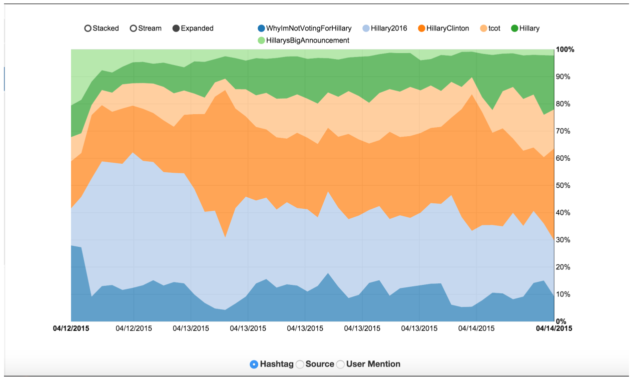


Figure 6. Expanded Graph

currently dominating the internet!

4.4. Show Reel

Show Reel graph is a series of animated visualization graph built in d3. We graph top 4 hashtags by time series in a variety of graphs, which includes:

- Lines
- Horizons
- Areas
- Stacked Areas
- Stream Graph
- Overlapping Areas
- Grouped Bars
- Stacked Bars
- Bars
- Donuts

Show reel is not the most informative graph of all, yet it is built with potentials to be broken down into individual graphs, and users (with very little programming backgrounds in JavaScript) can manipulate this graph easily for their purposes.

4.5. Word Cloud

Word Cloud, as one of the most popular and used graph for visualizing word count, fits greatly with our intentions of visualizing twitter data. Word cloud gathers the most mentioned words in both tweets and hashtags, and display them collectively, where the size of the word is proportionate to the count of the word in the dataset. As seen in Figure 7, ”president”, ”hillary2016” and ”campaign” are the most outstanding words/phrases amongst all, which conveys the main topics and trend directly to the viewers. Additionally, it is built with mouse-over function, where user can mouse over to a specific word, and exact count will be displayed.

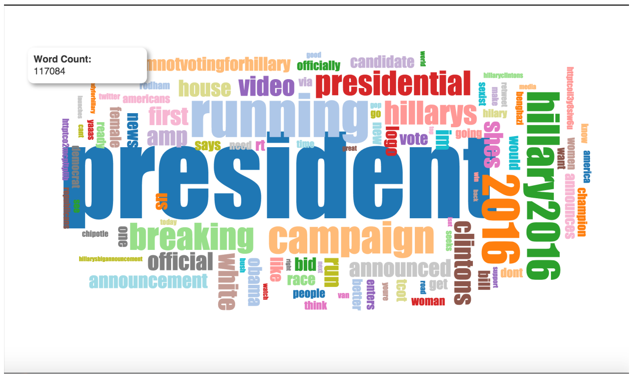
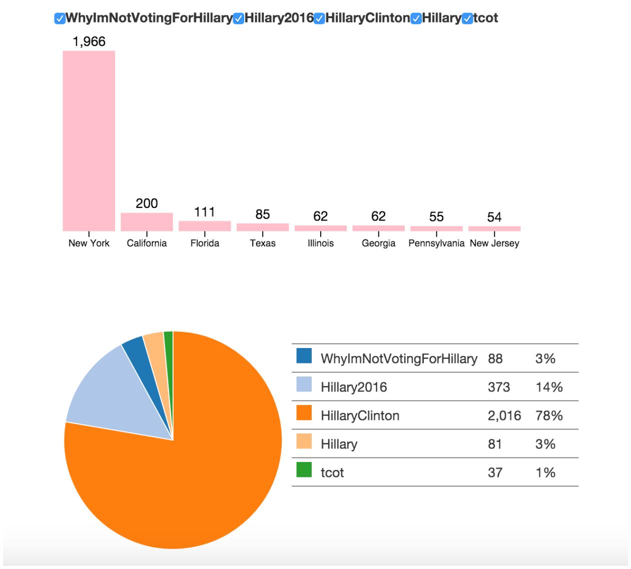


Figure 7. Word Cloud

4.6. Dashboard Graph

As the name suggests, this graph consists of 2 smaller graphs and a detailed legend. Dashboard graphs the top 5 hashtags by their frequences against geographical locations.

As in Figure 8, the dashboard consists of a pie chart, bar chart and an interactive legend. All parts of the graph interact with each other. Bars are counts by geographical locations, pies are counts by hashtag counts. When mouse-over to a specific bar, the pie chart will be updated accordingly to the information of the selected location. Similarly, all bars will be updated accordingly when mouse-over to a specific slice of pie chart (Figure 9), which equals to selection of a specific hashtag (color of bars will be updated to the color of the pie slice as well). We also add the checkbox function to avoid issues such as selections biases in dataset, where user can click off any hashtag (Figure 10) and all graphs and legend will be updated in-place. The checkbox labels are extracted from raw data and therefore correspond to the data passed in.



🟦	WhyImNotVotingForHillary	88	3%
🟦	Hillary2016	373	14%
🟧	HillaryClinton	2,016	78%
🟧	Hillary	81	3%
🟩	tcot	37	1%

Figure 8. Dashboard Graph

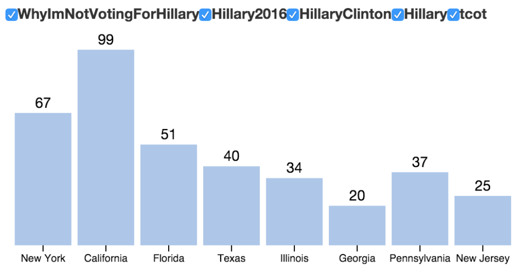
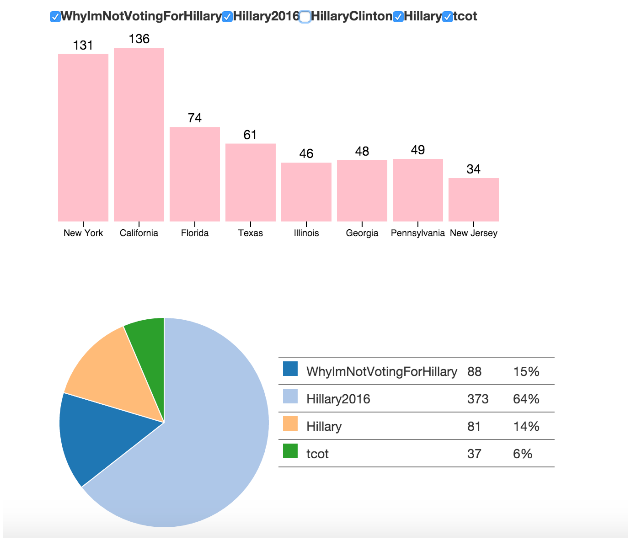


Figure 9. Tag selection on hashtags

4.7. Hierarchical Bundling Graph

Hierarchical Bundling Graph is designed for displaying the interactions among accounts in the dataset. Due to limited displaying space, we only include the top 150 accounts by the number of times they are mentioned. Mouse over to each account name can the user observe how and who this account is interacting with in the dataset. Green line means



🟦	WhyImNotVotingForHillary	88	15%
🟦	Hillary2016	373	64%
🟧	Hillary	81	14%
🟩	tcot	37	6%

Figure 10. Checkbox selections

that this account is mentioned by the account on the other end of the line, red line means that this account mentions the account on the other end of the line.

As the name suggests, accounts are grouped together by

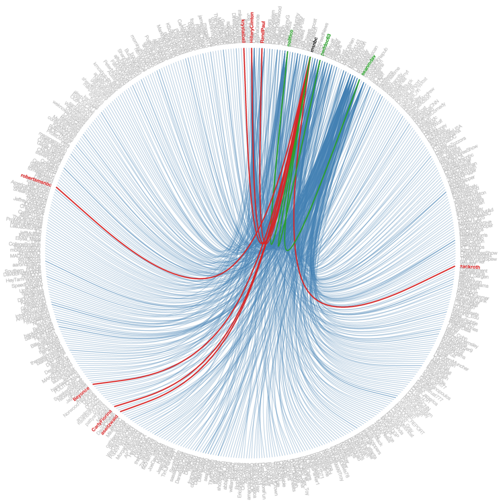


Figure 11. Hierarchical Bundling Graph

the number of time it's mentioned. In Figure 8, user can view the most mentioned accounts directly at the top of the graph, where most lines converge together. In this graph, political figures and news channels appeared the most, from which can the user infer the topics of all the tweets in the dataset (presidential announcement!). For individual who want to target at certain clientele or groups, this graph provides userful insights in making successful marketing strategic decisions.

4.8. Sunburst Graph

The last graph is the sunburst graph, where we analyze the semantic structures of the tweets.The graph is analogous to a series of concentric circles, where each layer represents the position of the word in the tweet. Due to graph function capacity, we only graph tweets up to the 10th word.

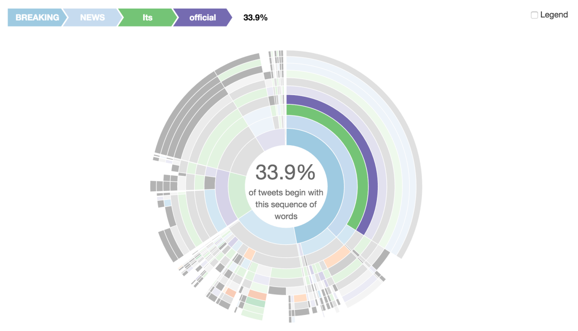


Figure 12. Sunburst Graph

The center of the graph indicates the percentage of all tweets that share the selected semantic structures, as well as the progress bar at the top of the graph. Sunburst graph is designed to grasp the key words/most mentioned phrases in all tweets, and analyze the various contexts in which they appear. Users can see both the positive comments and negative comments regarding Hillary Clinton's presidential announcements from this particular sunburst graph.

5. Future Directions and Potentials

We hope to achieve query selections within the dashboard, where each user can choose their own choices of fields for visualization. This is yet hard to accomplish for the following reason; Query selection requires either loading all data into JavaScript main functions, and extract in-place, or extract data of user's choice everytime from the whole dataset. Currently these two approaches are not executable because loading the entire dataset or wait for 8-10 min for every selection is impossible.

Second of all, it would be desirable if we could add more graphs to our current selections, therefore enable more comprehensive visualizations and better information retrieval. One of such example would be the choropleth graph, where we hope to project the geographical location of each tweets onto maps and visualize the activeness of tweeting activity by geographical location. However, the challenge we encounter (or most researchers who work with Twitter data encounters) is that most of the tweet locations are missing, and interpolating the locations can be tricky at times.

Thirdly, as we have been suggested about the possibility of Dahlia in, not just visualizing, extracting data from Twitter API. Digital Methods Initiative Twitter Capture and Analysis Toolset DMI-TCAT[6], which is a set of tools to retrieve and collect tweets from Twitter and to analyze them in various ways. DMI-TCAT has powerful functions to modify captured dataset and output in standard format, but does not provide much visualization.We currently consider this task to be out of the scope of the library functions, as extraction is not part of the definition of ”visualization”, and the unfortunate fact that extracting a significant amount of data from API can take up to days and is rather unlikely to come through in Dahlia.

Acknowledgments

Special thanks to Bowen Yu (yubowenok@gmail.com), PhD student of Professor Claudio Silva at Tandon School of Engineering, for his tremendous help to our project!

References

Dubois, P.F. Python: Batteries included. *Computing in Science and Engineering*, 9, 2007.

Hunter, J. D. Matplotlib: A 2d graphics environment. *Computing In Science & Engineering*, 9(3):90–95, 2007.

Langley, P. Crafting papers on machine learning. In Langley, Pat (ed.), *Proceedings of the 17th International Conference on Machine Learning (ICML 2000)*, pp. 1207–1216, Stanford, CA, 2000. Morgan Kaufmann.

M.Bostock, Ogievetsky, V., and Heer, J. D3: Data driven documents. *IEEE Transactions on Visualization and Computer Graphics*, 17, 2011.

Milmann, K.J. and Avaizis, M. Scientific python. *Computing in Science and Engineering*, 11, 2011.

[6]https://github.com/digitalmethodsinitiative/dmi-tcat