Final Report: NED's Vis

Michael Gottlieb *mikemgottlieb@gmail.com*
Emily Hindalong *ehindalong@gmail.com*
Dmitry Tebaykin *dmitry.tebaykin@gmail.com*

December 16, 2015

Abstract

Integrating neuroscientific data across labs is a huge challenge, NeuroElectro.org collects electrophysiology, neuron type and experimental conditions data from published articles and provides all of the data in text format for analysis. Here, we present NED's Vis - NeuroElectro Data's Visualization app that was developed in R-Shiny and deployed to neuroelectro.org. The tool allows direct interaction with the database skipping data download and wrangling steps, it includes 3 panels: overview, explore, filter. Overview panel provides insight into the amount of data behind each comparison type, where as explore panel is focused on direct comparisons and filter panel enables the user to eliminate uninteresting data from the analysis. Explore panel plots are highly interactive: plot type is defined by the types of chosen variables to display (quantitative versus qualitative), outlier removal is as easy as clicking the points off the plot, with optional zooming to a specific set of data points (each point is an article in the NeuroElectro database and on mouseover displays information needed to access it).

Introduction

Neuroscientists have conducted extensive research on the electrophysiological (ephys) properties of different neuron types, but there are barriers to comparing and aggregating results across different studies. This can be attributed to a lack of standard definitions and procedures as well as paywalls maintained by closed-access journals. To alleviate this, Tripathy et al.[1] have developed NeuroElectro - a freely available web-tool that allows users to directly compare data from different neuroscience articles. The primary goal is to "facilitate the discovery of neuron-to-neuron relationships and better understand the role of functional diversity across neuron types" [1].

NeuroElectro is a Django text mining and curation application http://neuroelectro.org developed mainly in C-Python with a javascript-based front end and an SQL database on the back end. It currently hosts experimental data from ~900 articles and is expected to grow to host the experimental data of thousands of articles. The data for each article can be accessed by the type of neuron, its electrophysiological properties, or via a table of articles.

However, the current visualizations provided by NeuroElectro are lacking: the user is only able to view data for one neuron type or one ephys property at a time using scatter plots. Our goal was to develop a new interface that supports seamless browsing and analysis of select subsets of the data.

Neuroelectro is a rare breed amongst text-mining projects in the fact that it allows end users to interact with curated data directly. Most text-mining tools in the biomedical domain assume that the end user will want an association matrix for terms in a controlled vocabulary, such as MEDLINE or MeSH terms [2],[3]. These tools automatically generate and output an association matrix without providing the user with a way to interface with the original data. This limits the analyses a user can perform.

Taking that flaw into account, we designed our NeuroElectro visualization mainly to provide the user with the overview of the gathered data and with the ability to explore it as they see fit. To accomplish said goals in the most efficient manner we performed an extensive literature search into online biomedical databases visualization efforts and the most efficient general ways to display as well as interact with the data similar to ours (on the scope of hundreds to thousands data rows with a few dozen variables).

Related Work

Solutions to Similar Problems

Neuroelectro provides some crude data visualizations in the form of static scatterplots and a single PCA analysis plot. Most text-mining tools in biomedicine do not use visualization at all, and those that do are restricted to analyses on the derived association matrix. For example, VOSviewer [4] uses colour and spatial position to visualize the semantic clustering and strength of association across text mined terms. The Trading Consequence project [5] focuses on mined trading documents supported by controlled vocabularies to generate maps of commodity trading over time.

Exploring relationships

Exploration of relationships between some of the properties in Neuroelectro's dataset is supported by the current version of Neuroelectro (Brain region vs Electrophysiology only, metadata is only accessible via database). However, it is a static set of strip plots that do not account for all combinations of variables and do not allow any interaction.

Exploration of relationships is a common task in many analytics platforms such as Tableau [6], SAP BOBJ AOLAP [7] and Microsoft Excel [8]. Generally, these platforms provide a tabular view of the data in addition to customizable visualizations to enhance users' exploration of the data. Our goal differs from these platforms as we are not including a tabular view, we are limiting the users' choices to provide a simpler experience, and we are using plots that are not easily achieved with these platforms (e.g. interactive plots, hive plot).

Providing an overview of data

Essentially, we are facing a problem of visualizing a network when we are trying to give an overview of our data. Over the years many solutions have been proposed for this type of task: hairball [9], matrix [10], arc diagram [11], call network [12], hive plot [13] are among the most common. Simply visualizing the network as a collection of nodes connected with edges (the hairball approach) seems impractical due to a large number of nodes and connections (currently: 150 nodes and ~10k edges), scaling is also a problem since the hairball only gets bigger with time. The matrix approach deserves some credit in terms of data visibility and it is a familiar visualization style to biologists, but there are 2 issues with utilizing matrices for this task:
1. Our data is 3 dimensional (neuron type, ephys. property, metadata) and 3D matrices are usually very hard to interpret, we could provide a faceted view of 1 matrix per metadata as a possible solution, but the amount of screen space that would require is enormous.
2. Matrices do not scale well, the labels get too small to be legible at some point. That said, a count matrix could provide a similar overview information to a hive plot as long as the number of parameters is small enough. In the implementation section we will discuss further our decision to use both a hive plot and a matrix approach.

A call network visualization would end up looking very similar to a hairball in our case, as a result we had to discard this possibility due to scalability issues. Arc diagrams came in as a close second as our visualization of choice - they are easy to interpret, pleasant to look at and they can scale reasonably well with the amount of evidence in the database. The problem with arc diagrams is that all nodes would end up being on one line and that does not represent the 3 distinct groups of nodes (neuron types, ephys. properties, metadata) in our data.

In the end, we decided to use hive plots for providing main overview of our data. Krzywinski, Birol, Jones and Marra [14] describe the advantages of hive plots in terms of gaining quantitative understanding when visualizing networks. They also support: multiple axes, information encoding in the nodes and edges, scaling. The one issue with hive plots is that they are a fairly new visualization style and researchers may have trouble understanding what they are looking at. However, we plan to provide a guide to interpreting the hive plot as well as provide links to the supporting literature. This feature would be on-demand and can be disabled in the application preferences.

Applications of Similar Solutions

Filter panels

The filter panel paradigm, where one panel is used to control what data appears in the main panel, is well established in visualization domain [15],[16],[17]. An alternative solution is the filter bar, which uses less screen real estate [18]. However, we have opted to stick with filter panels because they will never interfere with the main view and will make it easier for the user to track which filters are applied at any given time. Furthermore, the number of filtering options that we offer will require a larger section of the screen.

There are two basic attribute-based filtering paradigms: drill-down and parallel selection [19]. As the referenced blog post describes, Amazon uses drill-down filtering and Kayak uses parallel filtering. Our solution uses a hybrid of these, allowing the user to drill-down categories and apply parallel selection within. We intend to refer to this blog post when designing the specific details of our filter panel.

Connected scatterplots

Since Neuroelectro data is rather diverse (dozens of electrophysiology properties for each of over one hundred neuron types), we plan to utilize scatterplots [20] and connected scatterplots [21] for answering research-oriented questions. Haroz et al. showed the effectiveness of the latter in representing time-series data: even though connected scatterplots are novel to many users, they are excellent at being intuitive to understand and capturing and holding the viewer's attention.

Linked highlighting

There are a number of interaction approaches to linked highlighting in scatter plots [22]. Through our consultation with the stakeholder, linked highlighting on hover was emphasized as a critical element. However, this is not the only means of linked highlighting available. For example, linked brushing, where the user selects a subset of points to be highlighted, is a popular choice for multi-selection [22], [23].

Hive plot

Overviews of the data will make use of the work done by Krzywinski et al. [14] and Hanson [24] for our proposed hive plot. Various good examples of hive plot visualizations of networks are shown on the hive plot website [13]. We have developed our visualization based on the features that are most meaningful for our data and the questions we are trying to answer with the view (sparsity of data, node degree, edge weights, outliers and general trends in the data). We also use a matrix and a table approach to supplement the hive plot, since alone the hive plot does not provide enough details about connections between data types.

Colour

We have decided to use colour as a channel for linked-highlighting. Previous research suggests that colour is one of the most powerful visualization channels [25],[26] and, when used correctly, it can provide insight into the data so intuitively that the user wouldn't even need a legend to understand what kind of data the channel encodes for. On the overview panel colour is used as a measure of how many articles exist for each connection between nodes.

Data and Task Abstractions

Domain-specific data and tasks

NeuroElectro is a database of neuroscience articles and it applies text-mining as well as curation approaches to extract electrophysiology measurements, neuron type information and experimental setup conditions (metadata) from the html-encoded articles. At this point, text-mining alone is not reliable enough since neuroscientists authoring the articles in questions were not writing them using guidelines. As a result, each article is a snowflake of sorts - even with very well written algorithms automated text-mining is not at human text interpretation level yet. Hence the need for training undergraduate curators to verify the text-mining results and correct its errors. We focus on visualizing only the curated data, meaning that we have ~1000 articles worth of data. Note that not all articles contain all data types that NeuroElectro is able to store.

Domain-specific data types

1. Electrophysiology measurements
    - These are intrinsic neuron properties: membrane potential at rest, spike threshold (minimum membrane potential that causes a spike), input resistance, rheobase (minimum amount of current one needs to inject to cause an spike), etc.
    - Neurons communicate with the help of action potentials (voltage spikes) which are caused by cell's membrane voltage rising above the spike threshold causing a cascade of Sodium ions to flood into the neuron, propagating the action potential signal down its axon and to other neurons. The cell then closes Sodium channels and opens Potassium channels in order to return to its original state.
    - Neuron signalling is an electrochemical process and electrophysiology aims to record all meaningful characteristics that describe this process.
2. Neuron types
    - It is no secret that brain contains many different kinds of neurons. Neuroscientists have not decided on exactly how many neuron types there are and the debate has been ongoing for over a hunder years. Nevertheless, there are resources on the Internet that attempt to offer a classification for neuron types. NeuroElectro utilizes enhanced NeuroLex neuron classifications, eventually NeuroElectro may be offering its own neuron type hierarchy as we gather more data. For the purposes of our visualization, we distinguish 2 levels in the neuron type classification hierarchy - all neurons are assigned a brain region and a neuron type within that region. Each neuron type is assigned exactly one brain region (Neuron types that are present in many places or if their location is unknown comprise the "Other Region" brain region). These assignments were performed by a Dr. S.J. Tripathy - a neuroscience postdoc and the original developer of NeuroElectro.
3. Experimental conditions (metadata)
    - This data type stores information about the electrophysiological experiment itself, such as: species, strain, age and weight of the animal used, electrode type with which the measurements were taken, chemical solutions used to keep the brain slice moist and semi-alive, recording temperature, etc.
    - This data is important in order to compare ephys measurements from different experiments and labs.

NeuroElectro also stores data about each article: title, publication year, authors, etc.

Domain-specific tasks

1. Explore relationships between neuron types, ephys properties, and experimental conditions.
    - Find the neuron type, ephys measurement and metadata of interest.
    - View specific electrophysiology measurement for a specific neuron type (e.g. view rheobase values for Hippocampal CA1 pyramidal cells).
    - Compare ephys measurements across neuron types (e.g. resting membrane potential across all or a selected set of neuron types).
    - Explore the effect of metadata on an ephys measurement (e.g. action potential amplitude change with animal age).
2. Identify regions of the brain where particular neuron types are found.

3. Identify how many data points exist for different combinations of neuron types, ephys properties, and experimental conditions.
4. Find out how many articles support a specific analysis.
5. Summarize the data in the current analysis scope.
6. Lookup details for individual evidence lines extracted from articles.

Abstracted data and tasks

Abstracted data

Ignoring all domain-specific complications, we are dealing with 1 csv spreadsheet that contains ~1000 lines of ~150 variables. We can split these variables up into 4 types: quantitative measurement data, qualitative location data, mixed type explanatory variables and qualitative information about each line. The line information can be used for tooltips or tables to provide more context, but in itself, it is not interesting for the analysis.

Abstracted tasks

1. Explore relationships between three different data types.
   - Browse data available for analysis (data overview).
   - Identify distributions of quantitative attribute values for a particular categorical attribute values.
   - Compare distributions of quantitative attribute values for different categorical attribute values.
   - Identify correlations between quantitative attributes.
2. Navigate data type hierarchy to identify data of interest.
3. Identify how many data points exist for different combinations of data types.
4. Identify how many data points support a specific analysis.
5. Summarize the data in the current analysis scope.
6. Lookup details for individual data points (items).

Solution

> describe your solution idiom, analyze it according to the framework of the book, and justify your design choices with respect to alternative possibilities if you have done any significant algorithmic work, discuss the algorithm and data structures. You might choose to split out Interface into its own section

Implementation

R-Shiny app (general)

Our solution was built using the Shiny web application framework [27] for the R language [28]. Shiny server and ui components handle all transactions between the front and back end of our application. Each major component took advantage of a number of existing libraries, which is explained in more detail in the following subsections.

The first step of our applications performs data loading, cleaning and wrangling. Our app takes a csv dump of Neuroelectro's database as input. We load, clean and manipulate data using base R and dplyr [29] functions. At this point we also generate and cache or load the cached version of a modified dataset to speed up matrix view generation as the filters are changed. Once the data is loaded and prepped, it is passed to the ui and server components that house the core functionality of our app.

Navigation/filtering panel

The filtering panel uses collapse panels from the shinyBS package [30]. The Neuron Type and Organism trees use the shinyTree package [31]. The shinyTree source code was modified to improve appearance and introduce text wrap to long labels that encrouched on the space of other components. The shinyjs and V8 packages [32] & [33] were used to add JavaScript commands on startup to modify the shinyTree component's unruly behaviour. The filter options for continuous features use slider bars from the base Shiny framework. As with the shinyTree component, they were not perfectly suited for our needs. We modified the sliders to use log scales via JavaScript commands called via shinyjs and V8. We implemented how the filter states were applied to the data set and observers to update the plots only when the selected data had been changed. Default behaviour resulted in all plots being redrawn whenever a filter element was touched, even if its value was not changed (e.g. expanding a node on a filter tree or moving a slider without deselecting any data points).

Explore panel

The four plots of the explore panel share data that is filtered based on the state of the filter panel. The data displayed on each plot is determined by the two axis selectors above each plot. The axis selectors are standard Shiny ui components that did not require modification. The plots in the explore panel are generated dynamically depending on the type of data that the user selects to view. We used the ggvis package [34] to generate the plots in the explore panel as they promised easy interactivity. Our dataset changed based on filtering rules and axes selected and was passed to a function we implemented to determine what type of plot to show and to reduce repeated code. This lead to problems, as interactive ggvis features, such as hover and brush handlers, do not work with well or at all with dynamic datasets and inside functions. While we consider other plotting options, we have implemented an on click handler that highlights points in all plots in which they appear. We also implemented action buttons to clear highlighting, remove highlighted points from all plots and restore removed points.

Overview panel

The overview panel has three distinct components: The hive plot, the heat map and the table. The hive plot uses the HiveR package[[35]] as well as the RColorBrewer package [36] for colour selection. It also uses modified functions designed by an R blogger [37] for the hive plot data wrangling. The hardest part was to optimize hive plot data gathering (counting number of co-occurrences for each pair of ephys, brain regions and metadata). Initially, with a naive implementation it took about 2 minutes to generate and refresh the hive plot. Through a much more effective algorithm and built-in R co-occurrences counting functions the 2 minutes have been cut down to a couple of seconds. Nodes have been positioned identical distance apart on the axes for readability. Potentially, node distance could encode some information in the future. Node colour represents its degree, edge colour correlates with the number of articles supporting that edge. At first, we have encoded a few network properties into node position, size, edge width (e.g. node centrality, reachability, edge weight), however, we then realized that this information is not valuable and serves only to make the hive plot confusing.

The heatmap uses a derived dataset that contains whether or not each datum contains information regarding features of particular interest to the stakeholder. It computes an association matrix on demand which is passed to the pheatmap package [38] to display the associations and annotations. Like the hive plot, it uses the RColorBrewer package for its colour pallette.

The table is generated based on filter rules and is made entirely using the DT package [39] with some non-default parameters. As its title implies, it is simply a wrapper for the DataTables JavaScript library.

Heatmap, data table and hive plot respond to the filter panel, however, filtering and sorting done in the data table do not get reflected in the other views at this time.

Results

> should include scenarios of use and multiple screenshots of your software in action. walk the reader through how your interface succeeds (or acknowledge how it falls short) in solving the intended problem. if you did any evaluation (deployment to target users, computational benchmarks), do report on that here.

Discussion and Future Work

Originally, we planned to use javascript with a few libraries (D3.js [40], angular.js [41]) for our project, however, the team's familiarity with R and the desire to learn R-Shiny [27] played the decisive role in our framework choice. Given the chance to choose the framework again, we would have chosen R-Shiny due to its ability to handle large amounts of data and generate complex plots with relatively high speed. D3 has better integration with the front-end, but Javascript and CSS injections in R-Shiny serve a similar purpose, even if less elegantly.

We have learned that Shiny provides a solid framework for front-end web development and R data analysis visualizations. The dynamic integration of data frames and vectors in R with the rendered R-Shiny plots is quite impressive. Unfortunately, customizing the default behaviour becomes a choire because of the modular nature of Shiny. One could say that getting something to work in R-Shiny is fast and simple, but getting the exact feature to work (if it is not the default behaviour) can be very difficult and time consuming.

Data wrangling speed is a usual concern when dealing with large volumes of data that are transformed into plots. We ran into this problem with the overview panel - both the hive plot and the heatmap took a very long time to generate initially and we had to optimize their data gathering, parameter calculating and plotting features in order to achieve reasonable online app refresh speeds. The hive plot generation is already on an acceptable time scale, but the overview heatmap still takes too long - its generation pipeline can be optimized further if data wrangling is united with the hive plot's.

In the near future we would like to add more interactivity to the hive plot: each point and edge should provide a detailed tooltip upon mouseover or click, brush selection should allow to filter the data similar to the filter panel (and brush selection in the explore view). The explore view brush selection currently does not work - the user has to click each point that they want to remove individually (or use the filtering panel). The scatter plots in the explore panel still have the colour channel available, we could make use of that by allowing multi-variable plotting or splitting up one of the variables into groups.

Conclusions

We have successfully developed a newer and better visualization tool for the NeuroElectro website. R-Shiny framework ensures NeuroElectroVisuals's integratability with the NeuroElectro database and makes it capable of performing the main tasks defined by its original developer - Dr. S.J. Tripathy (outlined above in the data and tasks abstraction section). The new visualization provides an overview of the database with multiple plots and table as well as a more finely targeted exploration panel that allows neuroscientists to investigate the brain regions and electrophysiology properties of interest. The app also has a filtering panel that is integrated with both overview and explore panels and allowes subsetting the database by brain region, ephys values and experimental metadata. This functionality enables users to analyze the relevant data.

Our visualization can and will be improved to better serve field specialists. During the first few months of the beta deployment period we plan to gather usage data and feedback. With these we plan to ensure that we have covered every use case scenario and that the tool is running smoothly on various platforms. We are also hoping to get data about stress-resistance (online concurrent usage of the visualization app). If NeuroElectro becomes wildly successful and we have an absolute need for a more customized visialization, the possibility exists to re-write the existing code in javascript. Alternatively, more efficient R-Shiny implementation is also possible as not all the algorithms and functions have been optimized in the current implementation due to time restrictions.

References

[1] S. J. Tripathy, J. Savitskaya, S. D. Burton, N. N. Urban, and R. C. Gerkin, “NeuroElectro: A window to the world’s neuron electrophysiology data,” *Frontiers in neuroinformatics*, vol. 8, 2014.

[2] L. French and P. Pavlidis, “Informatics in neuroscience,” *Briefings in bioinformatics*, vol. 8, no. 6, pp. 446–456, 2007.

[3] D. Rebholz-Schuhmann, A. Oellrich, and R. Hoehndorf, “Text-mining solutions for biomedical research: Enabling integrative biology,” *Nature Reviews Genetics*, vol. 13, no. 12, pp. 829–839, 2012.

[4] N. J. Van Eck and L. Waltman, “Text mining and visualization using vOSviewer,” *arXiv preprint arXiv:1109.2058*, 2011.

[5] U. Hinrichs, B. Alex, J. Clifford, A. Watson, A. Quigley, E. Klein, and C. M. Coates, “Trading consequences: A case study of combining text mining and visualization to facilitate document exploration,” *Digital Scholarship in the Humanities*, p. fqv046, 2015.

[6] Tableau, *Tableau.* 2015.

[7] SAP, *SAP businessObjects analysis, edition for oLAP 4.0.* 2015.

[8] Microsoft, *Excel 2016.* 2016.

[9] M. Baitaluk, M. Sedova, A. Ray, and A. Gupta, “BiologicalNetworks: Visualization and analysis tool for systems biology,” *Nucleic acids research*, vol. 34, no. suppl 2, pp. W466–W471, 2006.

[10] N. Henry and J.-D. Fekete, “MatrixExplorer: A dual-representation system to explore social networks,” *Visualization and Computer Graphics, IEEE Transactions*, vol. 12, no. 5, pp. 677–684, 2006.

[11] S. Gaston, *Arc diagrams in r: Network visualizations.* 2013.

[12] K.-K. Yan, G. Fang, N. Bhardwaj, R. P. Alexander, and M. Gerstein, “Comparing genomes to computer operating systems in terms of the topology and evolution of their regulatory control networks,” *Proceedings of the National Academy of Sciences*, vol. 107, no. 20, pp. 9186–9191, 2010.

[13] M. Krzywinski, *Hive plot website.* 2013.

[14] M. Krzywinski, I. Birol, S. J. Jones, and M. A. Marra, “Hive plots—rational approach to visualizing networks,” *Briefings in bioinformatics*, vol. 13, no. 5, pp. 627–644, 2012.

[15] R. Bearavolu, K. Lakkaraju, W. Yurcik, and H. Raje, “A visualization tool for situational awareness of tactical and strategic security events on large and complex computer networks,” in *Military communications conference, 2003. mILCOM’03. 2003 iEEE*, 2003, vol. 2, pp. 850–855.

[16] M. Dumas, J.-M. Robert, and M. J. McGuffin, “Alertwheel: Radial bipartite graph visualization applied to intrusion detection system alerts,” *Network, IEEE*, vol. 26, no. 6, pp. 12–18, 2012.

[17] A. Sopan, A. S.-I. Noh, S. Karol, P. Rosenfeld, G. Lee, and B. Shneiderman, “Community health map: A geospatial and multivariate data visualization tool for public health datasets,” *Government Information Quarterly*, vol. 29, no. 2, pp. 223–234, 2012.

[18] N. Turner, *Why filter bars are better than left-hand filters.* 2015.

[19] G. Nudelman, *Best practices for designing faceted search filters.* 2009.

[20] M. Friendly and D. Denis, “The early origins and development of the scatterplot,” *Journal of the History of the Behavioral Sciences*, vol. 41, no. 2, pp. 103–130, 2005.

[21] S. Haroz, R. Kosara, and S. Franconeri, “The connected scatterplot for presenting paired time series,”

2015.

[22] S. Few, *Coordinated highlighting in context.* 2010.

[23] W. Chang, *Linked brushing in ggvis.* 2014.

[24] M. B. A. Hanson, “Package ‘hiveR’,” 2012.

[25] T. Munzner, *Visualization analysis and design.* CRC Press, 2014.

[26] V. Setlur and M. C. Stone, “A linguistic approach to categorical color assignment for data visualization,” *Visualization and Computer Graphics, IEEE Transactions on*, vol. 22, no. 1, pp. 698–707, 2016.

[27] W. Chang, J. Cheng, J. Allaire, Y. Xie, and J. McPherson, *Shiny: Web application framework for r.* 2015.

[28] R Core Team, *R: A language and environment for statistical computing.* Vienna, Austria: R Foundation for Statistical Computing, 2015.

[29] H. Wickham and R. Francois, *Dplyr: A grammar of data manipulation.* 2015.

[30] E. Bailey, *ShinyBS: Twitter bootstrap components for shiny.* 2015.

[31] Trestle Technology, LLC, *ShinyTree: JsTree bindings for shiny.* 2015.

[32] D. Attali, *Shinyjs: Perform common javaScript operations in shiny apps using plain r code.* 2015.

[33] J. Ooms, *V8: Embedded javaScript engine.* 2015.

[34] W. Chang and H. Wickham, *Ggvis: Interactive grammar of graphics.* 2015.

[35] B. A. Hanson, *HiveR: 2D and 3D hive plots for r.* 2015.

[36] E. Neuwirth, *RColorBrewer: ColorBrewer palettes.* 2014.

[37] V. M, *Network visualization – part 4: 3D networks.* 2013.

[38] R. Kolde, *Pheatmap: Pretty heatmaps.* 2015.

[39] Y. Xie, *DT: A wrapper of the javaScript library ’dataTables’.* 2015.

[40] M. Bostock, V. Ogievetsky, and J. Heer, “D3: Data-driven documents,” *Visualization and Computer Graphics, IEEE Transaction*, 2011.

[41] P. Darwin and P. Kozlowski, *AngularJS web application development.* Packt, 2013.