

Interactive Visualization and Event Detection in Time-series Data

by

Ngoc Huyen Nguyen

A Dissertation

In

Computer Science

Submitted to the Graduate Faculty
of Texas Tech University in
Partial Fulfillment of
the Requirements for
the Degree of

DOCTOR OF PHILOSOPHY

Approved

Dr. Tommy Dang
Chair of the Committee

Dr. Akbar Siami-Namin

Dr. Yong Chen

Dr. Yuanlin Zhang

Dr. Mark Sheridan
Dean of the Graduate School

August, 2023

Copyright 2023, Ngoc Huyen Nguyen

For my parents, who teach me how to learn.

ACKNOWLEDGMENTS

This dissertation comes to life thanks to the great contributions of my family, friends, colleagues, and mentors, who supported me throughout this journey.

I must first thank my advisor, Dr. Tommy Dang, whose expertise in data visualization provided the foundational knowledge for this work. Dr. Dang has been a vital mentor throughout my academic pursuits, who introduced me to the very first concept of visualization, showed me how meaningful visualization research could be, and challenged me to do my very best.

My warmest thanks to the wonderful professors of the Department of Computer Science at Texas Tech. A special thank you goes out to Dr. Akbar Siami-Namin for your incredible support and guidance from early on. Thank you to Dr. Susan A. Mengel and Dr. Yuanlin Zhang for your feedback and support all along; working with you over the last couple of years was a valuable learning experience for me to grow as a researcher and instructor. Thank you to Dr. Sunho Lim for setting an excellent example of an educator; being the TA for your class was one of the best things that happened to me as a grad student. Thank you to Dr. Yong Chen; you are always very kind, helpful and truly care about your profession and your students.

I am immensely thankful for my colleagues, Dr. Caleb M. Trujillo, Dr. Kevin Wee, and Dr. Kathleen A. Bowe, for the thoughtful support throughout my PhD journey. I am humbled by your encouragement and passion for education and qualitative research.

I am grateful for the inspirational educators whom I have the great luck to have met in life. Thank you to my undergraduate advisor Dr. Oanh Thi Nguyen, meeting you and learning from you have changed my life in the most meaningful way. My sincerest recognition goes out to Dr. Minh T. Nguyen, thank you for paving the way, believing in me, and motivating me to reach my full potential.

Thank you to my wonderful lab mates: anh Vinh, anh Vung, chi Ngan, anh Nhat, and Jake. Working alongside you has been a pleasure in the past few years, through thick and thin, through projects and seemingly endless manuscripts.

I am grateful for my amazing friends I met at Texas Tech, who have made my journey unforgettable. A special thank you to Apoorva, Lana, and Liyuan, for being by my side, whether it was the emergency room, stormy or sunny days. Thank you to Huixin and Moriom for your encouragement and the lovely boba tea dates. Thank you to Erin, Mary, and Lizzy, my roommates who have made our house a home.

Thank you to my best friends: Giang, who is in Germany, Hue, who is in Korea, and Hoa, who is in Vietnam. We are thousands of miles apart, but we don't let the distance

break our hearts. Your friendships inspire me to be a better person.

My special appreciation goes to Ms. Elizabeth Bowen and Dr. Jennifer Marciniak at the Texas Tech Graduate Writing Center. Thank you for your heartfelt encouragement and support and for organizing the exciting writing boot camps.

To myself when I first started my PhD program: Thank you for not giving up on me.

Above all, my deepest gratitude goes to my parents: my mom Hien Lan Thi Nguyen and my dad Tuan Van Nguyen, who provide me with unwavering love and support and always believe in me. Always. Who teach me meaningful life lessons and inspire my love for teaching. Who lead by example, showing me the value of kindness and the value of learning as a lifelong journey. A special thank you to my sister Minh Huyen, who is always there for me, shares memes with me and cheers me up no matter the occasion. To my extended family, thank you for loving and caring for this stubborn child.

To the family I am born with and the family I make along the way: thank you from the bottom of my heart. I am very lucky to have you in my life.

TABLE OF CONTENTS

DEDICATION	ii
ACKNOWLEDGMENTS	iii
ABSTRACT	viii
LIST OF TABLES	x
LIST OF FIGURES	xi
LIST OF ACRONYMS	xv
1 INTRODUCTION	1
1.1 Motivations	1
1.2 Research Contributions	2
1.3 Dissertation Organization	5
2 BACKGROUND AND RELATED WORK	6
2.1 Information Visualization	6
2.2 Interactive Data Visualization	7
2.3 Time-series Data and Temporal Events	9
3 WORDSTREAM: INTERACTIVE VISUALIZATION FOR TOPIC EVOLUTION IN QUALITATIVE DATA	10
3.1 Introduction	12
3.2 Related Work	13
3.2.1 Word Cloud Models	13
3.2.2 Time-series Visualizations	14
3.2.3 Combined Models	15
3.2.4 Opportunities for Text Stream Visualization	15
3.3 Methodology	16
3.3.1 WordStream Design Decisions	16
3.3.2 WordStream: Computing The Visualization	17
3.3.3 WordStream Library	19
3.3.4 WordStream Maker Platform	22
3.3.5 WordStream Maker Architecture	23
3.4 Results and Discussion	24
3.4.1 WordStream Case Study: Exploring IEEE VIS Author Contributions	24
3.4.2 Informal User Study	24
3.4.3 WordStream Quantitative Evaluation	26

3.4.4	WordStream Maker User Interface	27
3.4.5	WordStream Maker Case Studies	28
3.5	Conclusion	31
4	DEVELOPING QUALITATIVE DATA DASHBOARDS: CASE STUDIES WITH WORDSTREAM	33
4.1	Introduction	34
4.2	Related Work	35
4.3	Methodology	37
4.3.1	Case Studies: Descriptions	37
4.3.2	Data Processing	39
4.3.3	Design Decisions	40
4.4	Results and Discussion	41
4.4.1	Case Study 1: Journal Data Dashboard	41
4.4.2	Case Study 2: Earthquake Situational Analytics	47
4.4.3	Informal User Study	53
4.5	Conclusion	55
5	QUANTITATIVE DATA DASHBOARD WITH MALVIEW: INTER-ACTIVE VISUALIZATION FOR COMPREHENDING MALWARE BEHAVIOR	57
5.1	Introduction	58
5.2	Background and Related Work	60
5.2.1	Signature-based Features/Static Analysis	61
5.2.2	Image-based and Dynamic Analysis	62
5.2.3	Hybrid (Static and Dynamic) Analysis	63
5.2.4	Visualization Tools and Analysis	64
5.3	Methodology	66
5.3.1	Capturing Dynamic Behavior Using ProcMon	66
5.3.2	MalView: System Overview	67
5.4	Results	70
5.4.1	MalView: Visual Components	70
5.4.2	Case Studies	75
5.5	Discussion	86
5.5.1	The Influence of Running Platforms on Malware Behavior	86
5.5.2	Malware Visualization Tools vs. MalView	89
5.6	Conclusion	93
6	AUTOMATED EVENT DETECTION FRAMEWORK	95
6.1	Introduction	95
6.2	Related Work	97
6.3	Methodology	100
6.3.1	Data Transformation	100
6.3.2	Time-series Event Formulation	101
6.3.3	Network Event Formulation	102

6.3.4	User-specified Inputs	104
6.3.5	Design Guidelines	104
6.4	Results	105
6.4.1	Case Study with EQSA	106
6.4.2	Case Study with MalView	110
6.5	Conclusion	112
7	CONCLUSION	113
7.1	Review of Research Contributions and Impacts	113
7.2	Limitations and Future Directions	114
7.3	Concluding Remarks	115
	BIBLIOGRAPHY	116

ABSTRACT

Visualization enriches human understanding of data through visual representations across many levels, from solving domain-specific tasks to addressing generalizable problems across disciplines. To gain a comprehensive view of visualization practice in time series, this dissertation delves into two general data types: qualitative and quantitative. For each data type, we examine how visualization helps solve domain problems and the implication of such solutions in the broader context. By combining different visualizations into one coordinated view, multiple aspects of data can be shown in conjunction with one another. To this end, we present three case studies of qualitative and quantitative visualization dashboards and close with an overarching event detection framework that can be generalized beyond the specific scenarios presented in the case studies. This dissertation contributes novel visualization designs and techniques that foster interactive systems for exploratory analysis of time-series data.

From the *qualitative time-series visualization* perspective, the field witnessed an increasing need for a method that could effectively demonstrate the progression of topics without losing the relevant context. As a result, **WordStream** was created to help visualize topic evolution while providing a visually appealing representation of qualitative text data. WordStream emphasizes frequent topics that emerge from the text source, especially when they fluctuate significantly. As a natural extension of WordStream to make it more applicable to a broader demographic, we developed **WordStream Maker**, an end-to-end platform that automatically generates WordStream visualizations with input from raw text data. This development makes it possible for users without prior programming knowledge to create WordStream with ease. We then employed WordStream in a multiplicity of contexts to explore its potential, constraints, and adaptability in different situations, which were grounded in two case studies: **Earthquake Situational Analytics (EQSA)** from social media and **Journal Data Dashboard** from educational assessments. The findings demonstrated that WordStream was intuitive, clear, and easy-to-use to explore text entries, especially words of interest. The potential of this tool can be extended for larger real-world scenarios such as text analysis of longitudinal studies.

To explore the *quantitative time-series visualization* realm, we proceeded with a typical case of numerical log data. Recognizing the need to leverage visual representations in malware analysis, we developed **MalView**, an interactive visualization platform for comprehending malware behavior. This tool provided a comprehensive visualization dashboard to represent the behavioral properties of malware classes, aiming to capture the common visual signatures of these malicious applications. Several case studies

showed that MalView offered additional features and information compared to several existing visualization tools to facilitate the malware analysis process, including temporal dependencies and interactions among processes. The platform was designed to offer scalability to multiple malware families and provide a useful asset for malware experts.

From the lessons learned from building qualitative and quantitative visualization dashboards, we formulated an **Automated Event Detection Framework** that supports identifying temporal events automatically. While events can be discerned through visual inspection, the escalating complexity of data introduces challenges that necessitate automated analysis techniques. The framework utilized the underlying graph structure from time-series data to examine the relationships among entities and extract the associated event features. Based on these identified characteristics, temporal events were detected. To evaluate the effectiveness of this method, we returned to the previously presented qualitative and quantitative dashboards to apply the framework and test its usefulness and practicality. The findings demonstrated that automatic detection mechanisms supported identifying complex events beyond immediate observation, offering a potential solution to alleviate the cognitive load associated with manual visual inspection.

LIST OF TABLES

5.1	A feature-based comparison of Anyrun [14], Hybrid [191] and MalView . .	92
6.1	Structural properties of a network structure commonly encountered in visualization and visual analytics systems.	98
6.2	Overview of Temporal features in network evolution (Ahn et al. [4]). . . .	99

LIST OF FIGURES

1.1	Overview of the work in this dissertation. We view qualitative and quantitative data visualization through the lens of visualization techniques, multiple-view dashboards for domain problems, and a generalizable framework.	3
3.1	WordStream visualization and WordStream Maker pipeline.	11
3.2	A standard word cloud and a stream graph are synthesized into a single snapshot (on the right).	17
3.3	The main components of our WordStream visualization: Preprocessing data, building boxes, placing terms, and drawing.	18
3.4	Building boxes and placing terms into stream layers.	18
3.5	Term selection in the WordStream layout: <i>boston</i> (location) and <i>google</i> (organization).	19
3.6	An example of how to import WordStream library to visualize data.	20
3.7	A snippet of the input JSON file.	21
3.8	The resulting WordStream visualization from the above program and data in Figures 3.6 and 3.7.	22
3.9	Popular IEEE VIS authors over 10 years from 2007 to 2016: author names are colored by their first publication venue.	25
3.10	Comparisons of the Compactness measure for 10 test data sets (from left to right).	26
3.11	Visual interface of WordStream Maker.	28
3.12	Journal data for education assessments throughout nine weeks. Different combinations of Part-of-speech (POS) Tagging, Named-entity Recognition (NER) in the NLP module, along with Frequency and Sudden Change in the text characteristics module.	30
3.13	Keywords extracted from publications on protein pathways from 1996 to 2014. Different combinations of Part-of-speech (POS) Tagging and Named-entity Recognition (NER) with frequency.	31
4.1	The schematic overview of the process to produce the interactive tool for visualizing formative educational assessment data. The visualization provides topics within contexts, user interactions with linked views and supports interactive features, such as highlighting, filtering and sorting	37
4.2	The topic summary view. The word selection from the word cloud includes: <i>research</i> , <i>question</i> (noun), <i>think</i> , <i>code</i> (verb), <i>different</i> , <i>specific</i> (adjective), resulting in the highlighted corresponding terms in the details. The word <i>data</i> is highlighted in the word cloud upon mouse-over interaction.	42

4.3	WordStream visualization: A time-series visual representation of the evolution of topics of interest. The visual components: (1) Journal prompt reference to weekly journal questions, (2) WordStream visualization for topic evolution, and (3) Students' responses section that helps explore into details.	44
4.4	User interactions on WordStream view. There are several interactions are at play here. First, the words " <i>learn</i> " and " <i>data</i> " are clicked and the two corresponding streams representing the changes in their frequency are shown. The students' responses are filtered to show only the records containing " <i>learn</i> " and " <i>data</i> ". Second, the word <i>easy</i> is mouse-overed, hence its occurrences in other time points are highlighted. Third, the search input is <i>visualization</i> , thus the responses are filtered again to only keep the records containing <i>visualization</i>	46
4.5	EQSA Earthquake Situational Analytics dashboard interface. The main control panel (A) is built as a stacked area chart, showing the volume of messages according to chosen categories from the selection panel (A1). For the chosen time frame and the chosen categories: Panel (B) presents topic evolution, panel (C) is a map in which each neighborhood's color indicates the number of posts, panel (D) is a network user interaction, and panel (E) is a chart for ranking content creators. Interactive features are provided to support the analysis, including a sliding window for time frame selection with <i>adjustable</i> window size, mouseover events providing detail-on-demand messages (at the black arrows). In panel (A), three high red peaks correspond to three times the earthquake strikes. Three panels at the top of this figure show the filtering option corresponding to "Resource" instead of "Event" as in the bordered panels below.	47
4.6	Three times the earthquake strikes (upper panel) and details for first strike (lower panel). Details-on-demand with WordStream shows seismic event messages.	49
4.7	(a) Event of flooding; (b) Detail of events that happened in the city and their corresponding locations; (c) Trending of flooding event concerns over time.	50
4.8	(a) Event of fire; (b) Detail of events that happened in the city and their corresponding locations; (c) Trending of fire event concerns over time. . . .	51
4.9	Emergency Manager (left) and dereknolan (right), two accounts that have influence in the community.	51
4.10	Five hours after the third strike, the main problem comes from "bridges" and "route" (terms emphasized in the WordStream), with multiple notifications from the Department of Transportation of St. Hi-mark. © 2019 IEEE.	52

5.1	The schematic overview of MalView framework for analyzing the dynamic behavior of malware. The visualization provides linked views and supports interactive features, such as filtering, highlighting, ranking, and details-on-demand.	68
5.2	The MalView system contains the following main views: <i>Input and Operation Overview</i> (A), <i>Processes Activity</i> (B), <i>Classification</i> (C), <i>Process Dependencies</i> (D) and <i>Libraries Matrix</i> (E). The <i>Process Activity</i> view has another mode of showing the <i>referenced operated object</i> (B1), with feature of lensing for zooming in upon a particular interval in both viewing modes. Detail-on-demand is provided upon mouse interaction on selected item, as in the example in panel (B2).	70
5.3	MalView analysis summary of TeeracB malware on Windows 7.	73
5.4	MalView analysis summary on RAT, with the filter set on displaying the interactions with RAT only: There are multiple and repeated function calls between the process corresponding to the RAT and <i>explorer.exe</i> , <i>wscript.exe</i> and <i>svchost.exe</i> . Patterns of periodical operations are also presented, such as of Local Security Authority Subsystem Service <i>lsass.exe</i> or Virtual Box's <i>vboxservice.exe</i>	77
5.5	MalView visualization for a sample Trojan. User can request to overlay the networks of suspicious processes (which contain the self calls).	78
5.6	Trojan MultiInjector under MalView analysis: the sequence of Process Create events generated by the malware and its interactions with other processes in the system.	80
5.7	MalView visualization of malware Backdoor Androm on Windows 7. The top area chart demonstrates two separate phases: the first one with malware activities, ending with a call to <i>regasm.exe</i> (Box A) and the second with recurring patterns involving <i>regasm.exe</i> itself (Box B).	81
5.8	MalView analysis summary of ransomware WannaPeace on Windows 7.	82
5.9	MalView shows suspicious activities from a Behavior malware, named <i>Bladabindi</i> . Panel (a) shows that it starts <i>netsh.exe</i> to modify firewall configuration. Panel (b) depicts that it sets the registry values to run itself.	84
5.10	MalView view on a Hacktool malware type called <i>Mailpassview</i> . It first creates process <i>svchost.exe</i> (a), then <i>svchost.exe</i> starts <i>windows update.exe</i> (b), and then <i>windows update.exe</i> creates <i>pidloc.txt</i> (c)	85
5.11	The classification for malicious connecting domains from malware <i>Mailpassview</i> . The target domain <i>iplogger.com</i> is assessed as malicious and suspicious by VirusTotal. This domain is connected from <i>powershell.exe</i> via four activities: <i>TCP Connect</i> , <i>TCP Send</i> , <i>TCP Receive</i> and <i>TCP Disconnect</i>	85
5.12	MalView visualizations of Email Flooder on (a) Windows XP, (b) Windows 7, and (c) Windows 10.	86
5.13	<i>Bladabindi</i> malware executions on different platforms: Windows XP (a), Windows 7 (b) and Windows 10 (c).	88

5.14	Patterns of malware behavior across platforms: <i>Bladabindi</i> malware executions on (a) Windows XP, (b) Windows 7, and (c) Windows 10, all under focus and context technique with busy interval length of 20 seconds. On the right end are the corresponding dependency graphs of the malware process. While different processes express different dependency graphs, as shown in Figure 5.5 and Figure 5.7, the dependency graph of <i>Bladabindi</i> malware is relatively consistent across different platforms.	89
6.1	Overview of the proposed automated event detection framework.	100
6.2	Common shapes encountered as events in time-series analysis.	102
6.3	The space of network evolution tasks in three dimensions: Entities, Properties, and Temporal Features. Each point in this space represents an element in the 3-fold Cartesian product from the three dimensions. For example, with Temporal feature as <i>Growth</i> , Structural property as <i>Betweenness Centrality</i> , and Entity as <i>Node</i> , we have the task of examining the growth of betweenness centrality of a node in a network. This task can then be translated as the growth over time of the influence a node has over the flow of information in a graph.	103
6.4	Original visual interface of the Earthquake Situational Analytics [157]. We will focus on the two highlighted panels for Timeline-based visualization and Network.	106
6.5	Temporal peak detection with $granularity_{spike} = 13$ and $threshold_{ratio} = 6$. Boxes 1, 2, 4, and 5 are verified and kept for further exploration.	107
6.6	Top 5 biggest network growth of centrality at $granularity_{growth} = 20$	108
6.7	Details into the largest growth of 1.2: After the first peak found in Box 1, the network was scarce but then grew significantly and formed four major clusters, hence the increase in centrality.	108
6.8	Highest speeds of changes were detected at $granularity_{speed} = 17$. Negative signs mean the amounts of changes decline; e.g., the zoomed-in interval shows a significant decrease to zero of centrality.	109
6.9	Top 5 of the corresponding acceleration of centrality of the network at the same $granularity_{speed} = 17$	109
6.10	An example of dependency graph with with malware <i>bladabindi.gen</i> and its shared resources with <i>netsh.exe</i> . Figure (a) shows all nodes in the compact form and Figure (b) shows the expanded forms of two corresponding nodes of type File and DLL in (a). On the expanded view (b), edges are depicted explicitly.	110
6.11	Detection results with malware <i>bladabindi.gen</i> . The upper panel shows the process activity view of MalView, and the lower panel shows a detailed view of the growth of edge creation per second. The two peaks are detected, marked as A and B, with their corresponding dependency graphs.	111

LIST OF ACRONYMS

DOM Document Object Model

EQSA Earthquake Situational Analytics

NLP Natural Language Processing

POS Part-of-speech

CHAPTER 1

INTRODUCTION

Why do we visualize data? Because our visual perception is incredibly powerful at recognizing patterns and identifying structures in data. It's not a coincidence that when we understand something, we say, "I see."

– *Robert Kosara*

1.1 Motivations

Data visualization is first and foremost based on data. By mapping data to visual properties, visualization is designed to help us get a better understanding of data. As human, our visual perception has a powerful capability to identify patterns and discern structures within data [115]. By offloading cognitive processing to the perceptual system, the natural abilities of humans to detect patterns visually can be leveraged [97]. This power of visualization assists us in reaching multiple goals, including exploring new datasets and discovering patterns (*exploration and analysis*), effective communication of results and relationships in data (*explanation*), and guiding our audience through an argument and supporting their decision-making process (*presentation*).

While there are many interesting visualization topics out there, this dissertation purposefully focuses on time-series data and the evolution of data patterns over time. There are two reasons for this direction. First, studying time-series visualization helps us make sense of the immediate world around us, which can present itself as a temporal sequence of a myriad of events simultaneously happening. Thinking about time helps us reflect on the changes in life and our own individual growth. Second, by studying the visualization design for time series, we realized that there is a *robust transferability* of the visual representation that is applicable for other disciplines that share the linearity characteristic, such as the sequence of amino acids in each protein or the physics of sound waves. It is important to note that a time-series dataset is typically an ordered sequence of time–value pairs, which are often but not always spaced at equidistant temporal intervals [151, 116].

Where do we begin when we visualize data? According to the nested model for visualization design and validation proposed by Munzner [150], the first stage is to characterize the task and data in the vocabulary of the problem domain. That understanding is then translated into operation and data type abstraction within the vocabulary of

information visualization, followed by the design of visual encoding and interaction. The innermost level is the creation of an algorithm to execute techniques efficiently. This model depicts the capability of visualization to facilitate the understanding of data through visual representations across many levels, from solving domain-specific tasks to addressing generalizable problems across disciplines. Furthermore, domain problems inspire the development of visualization designs that exhibit transferability across different domains, shedding light on the broader significance and implications.

To gain a comprehensive view of visualization practice in time series, this dissertation delves into two general data types: qualitative and quantitative. For each data type, we examine how visualization helps solve domain problems and the implication of such solutions in the broader context of exploratory analysis. Leveraging multiple coordinated views with dashboard design, different facets of data can be presented in conjunction with one another. For this purpose, we propose three case studies of qualitative and quantitative visualization dashboards and close with an overarching event detection framework that can be generalized beyond the specific scenarios described in the case studies. Throughout this study, we discuss lessons learned in developing interactive time-series visualizations from domain-specific perspectives, building an automated framework for temporal events, and bridging the gap between qualitative research and data visualization.

1.2 Research Contributions

This dissertation contributes novel visualization designs and techniques that foster interactive systems for exploratory analysis of time-series data. The contributions are depicted along with corresponding chapters in Figure 1.1.

An interactive technique for visualizing topic evolution in text data.

From the *qualitative time-series visualization* perspective, the field witnessed an increasing need for a method that could effectively illustrate the progression of topics without losing the relevant context. We created **WordStream** [49] (Chapter 3) to bridge this gap. This approach is helpful for visualizing topic evolution while providing a visually appealing representation of qualitative text data. WordStream emphasizes frequent topics that emerge from the text source, especially when they fluctuate significantly. Since it was published in 2019, WordStream has been applied in multiple scenarios, from social media content analysis to qualitative analysis in educational assessments to explore the evolution of text data over time. The tool has been packaged as an open-source library in JavaScript [154].

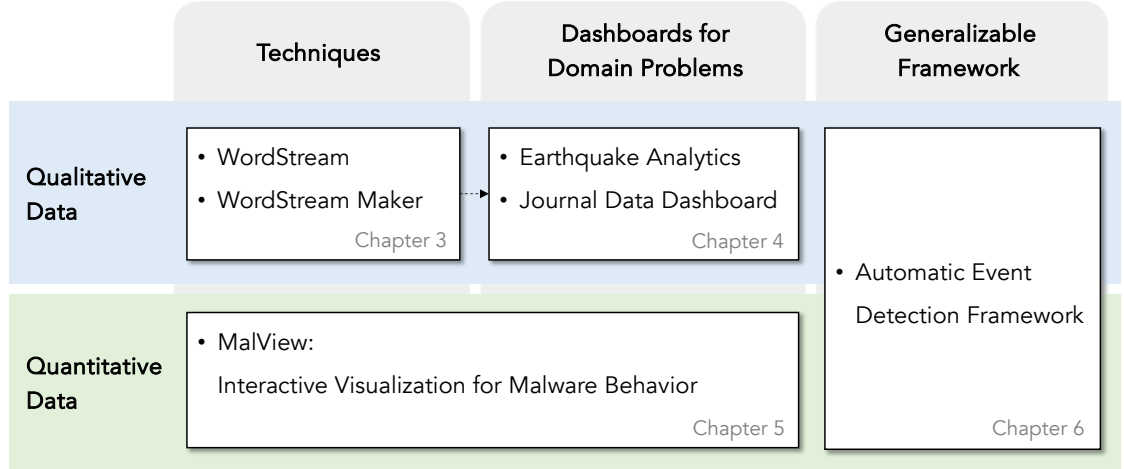


Figure 1.1: Overview of the work in this dissertation. We view qualitative and quantitative data visualization through the lens of visualization techniques, multiple-view dashboards for domain problems, and a generalizable framework.

A lightweight end-to-end platform for creating WordStream visualization.

Recognizing the potential of WordStream visualization, we saw an opportunity to make it more applicable to a broader demographic.

By combining the capabilities of both text and visualizations, WordStream has previously made exploring topic evolution in text data easier. However, the amount of data wrangling required to produce a WordStream remains a significant barrier for non-technical users. As a natural extension of WordStream, we developed **WordStream Maker** [159] (Chapter 3), an end-to-end platform with a pipeline that uses Natural Language Processing (NLP) to assist non-technical users in processing raw text data and generating customizable visualizations, all without the need for programming expertise. This development made it possible for users without prior programming knowledge to create WordStream with ease. To date, the open-source repository of WordStream Maker has more than 30 stars on [GitHub](#).

Qualitative time-series visualization dashboards: Case studies to demonstrate how WordStream can be employed to explore qualitative data.

Proceeding with the qualitative structure, we employed WordStream in a multiplicity of contexts to explore its potential, constraints, and adaptability in different situations. The approach was grounded in two case studies focusing on qualitative data (Chapter 4). In our first case study, we built an Earthquake Situational Analytics (EQSA) [157] dashboard that helped amplify the concerns of the community during a series of earthquake-related events. From the identified community needs, resource allocation and assistance were provided. The second case study presented **Journal Data Dashboard** [161] to

leverage educational journal entries to discover the patterns and diversity of student ideas in a context-specific approach. The findings demonstrated that WordStream was intuitive, clear, and easy-to-use to explore text entries, especially words of interest. The potential of this tool can be extended for larger real-world scenarios such as text analysis of product reviews or longitudinal studies.

Quantitative time-series visualization dashboards: An interactive platform for exploring malware behavior via case studies focused on different malware families.

To explore the *quantitative time-series visualization* realm, we proceeded with a typical case of numerical log data. Understanding the value of incorporating visualizations in the field of malware analysis, we developed **MalView** [155] (Chapter 5), an interactive visualization platform for comprehending malware behavior to ease the analysis process. There are two popular techniques currently used for analyzing given software, which are effective in detecting certain classes of malware. In static analysis, the malicious program is parsed, and representative artifacts are produced without execution, whereas dynamic analysis requires the execution of the malicious application. By leveraging the visual representation of the artifacts created by both static and dynamic analysis tools, the proposed comprehensive visualization dashboard details the behavioral properties of major malware classes. This approach aims to capture the common visual signatures of these malicious applications. Case studies showed that MalView offered more features and information compared to several other visualization tools, including facilitating the malware analysis process by examining temporal dependencies and interactions among processes. The platform was designed to offer scalability to multiple malware families and provide a supplementary asset for malware experts.

An automatic event detection framework for time-series data.

From the lessons learned from building qualitative and quantitative visualization dashboards, we formulated an **Automated Event Detection Framework** (Chapter 6) that supports identifying temporal events automatically. Although events can be identified through visual inspection, the increasing complexity of data poses challenges that call for automated analysis techniques. The framework utilized the underlying graph structure from time-series data to examine the relationships among entities and extract the associated event features. Based on these identified characteristics, temporal events were detected. To evaluate the effectiveness of this method, we returned to the previous qualitative and quantitative dashboards to apply the framework and test its usefulness and practicality. The results indicated that automatic detection mechanisms facilitated the identification of complex events beyond immediate observation, presenting a potential solution to alleviate the cognitive effort required for manual visual inspection.

1.3 Dissertation Organization

The visualization designs and techniques we propose in this dissertation address different challenges that a data analyst may encounter when dealing with time-series data. Figure 1.1 provides an outline of the structure and the underlying perspective upon which this dissertation is constructed. This dissertation explores two fundamental categories of time-series data: qualitative and quantitative. We will examine domain problems for each branch, map them to the vocabulary of visualization in terms of data and tasks, and design the algorithms and visual encodings under multiple-view visualization.

With qualitative data, we first built WordStream technique to visualize topic evolution and its platform extension, WordStream Maker. Details of these developments are elaborated in Chapter 3. We created the technique as a standalone tool before incorporating it into larger scenarios. The next chapter, Chapter 4, explores WordStream’s adaptability in different situations, which were characterized in two case studies: EQSA from social media and Journal Data Dashboard from educational assessments. While the former employed WordStream as a visualization panel along with other views, the latter emphasized WordStream as a primary focus to guide the analysis.

With quantitative data, Chapter 5 explores different facets of numerical log retrieved from process monitoring. New techniques are introduced to demonstrate the interaction among system processes and dependencies, with the ultimate objectives of providing thorough comprehension of malware behavior and easing the malware analysis process. Chapter 6 closes the two branches of qualitative and quantitative exploration with an overarching event detection framework that can be generalized beyond the specific scenarios outlined in the case studies.

CHAPTER 2

BACKGROUND AND RELATED WORK

This chapter situates the study in this dissertation in the context of a larger community of scholarship. We present an overview of the area of time-series analysis in the field of data visualization, identifying the existing gaps, exploring potential solutions, and highlighting the driving inspirations behind this study.

2.1 Information Visualization

The origin of information visualization can be traced back to cartography and astronomy. In the 1st century AD, Claudius Ptolemy crafted a spherical projection map of the Earth using latitude and longitude to characterize position, establishing a reference standard until the 14th century. The earliest known attempt to show quantitative information was around the year 950, illustrating the changing values of the positions of the sun, moon, and planets throughout the year [75]. However, it was not until the 17th century that the French philosopher and mathematician René Descartes invented a two-dimensional coordinate system to display quantitative data graphically [17]. Following Descartes' innovation, the late 18th and early 19th centuries witnessed notable progress in graph creation. William Playfair, a Scottish social scientist, made substantial contributions by inventing or significantly improving many of the graphs we use today, including bar charts and pie charts [74]. In 1858, Florence Nightingale, one of the most recognized women of her time as the founder of modern nursing, used circular area charts to indicate that more British soldiers had died during the Crimean War as a consequence of poor hygienic conditions in battlefield hospitals than in combat [165].

In modern data visualization practices, existing literature has cultivated theoretical frameworks on visualization analysis and design. In 1977, John Tukey introduced *Exploratory Data Analysis* [209], a predominantly visual approach to exploring and analyzing data, about “looking at data to see what it seems to say”. The principle behind this approach is to examine the data before applying a particular probability model. In 1983, Edward Tufte made a significant contribution to the field with his revolutionary book titled *The Visual Display of Quantitative Information* [208], showing us the effective ways of displaying data visually, with “clarity, precision, and efficiency.” In terms of formalisms, Leland Wilkinson’s *Grammar of Graphics* stands as one of the most influential works for specifying statistical graphics [224], where a grammar of graphics

is a tool that enables us to concisely describe the components of a graphic [223]. A more recent work by Munzner, *Visualization Analysis and Design* [151], offers a broad synthesis of principles and guidelines for the design and analysis techniques.

The advancements in computer technology have had a profound impact on the way data is processed, analyzed, and visualized. Advanced algorithms and computational power enable efficient data management [29], aggregation [70], and transformation [201] necessary for effective visualization. With the increasing availability of real-time and streaming data sources, there have been significant developments in visualization techniques that can handle and visualize data as it arrives. Real-time visualization enables the monitoring and analysis of high-performance computing [50, 51], network behaviors [23], biophysics and computational biology [179], and nanoparticle clustering [124], to name a few. Visualization tools and libraries enable users to generate, publish, and share visualizations, making visualization more accessible. The *ggplot* library [222] implemented a variant of Grammar of Graphics in the language R. Some notable examples of visualization tools and libraries include D3.js (Data-Driven Documents) [26], Vega-lite (High-level visualization grammar) [188], Gosling (Grammar for interactive and scalable genomics data visualization) [128], and Tableau – the commercialized version of Polaris [201].

2.2 Interactive Data Visualization

This dissertation is motivated by two visualization mantras that govern exploratory visualization with interactivity. First is the Visual Information Seeking Mantra [197]:

“Overview first, zoom and filter, then details-on-demand.”

– *Ben Shneiderman, 1996*

Later, Keim adjusted the mantra to shift its focus towards Visual Analytics, emphasizing the importance of analytical approaches combined with advanced visualization techniques [111]:

“Analyze first, Show the Important, Zoom, filter and analyze further, Details on demand.”

– *Daniel Keim, 2008*

Interactive data visualization refers to exploring and analyzing data directly within the visualization itself using interactivity, enhancing user engagement, and facilitating

data exploration. In a comprehensive work on interaction for data visualization [63], Dimara and Perin have defined interaction as follows: “Interaction for visualization is the interplay between a person and a data interface involving a data-related intent, at least one action from the person and an interface reaction that is perceived as such.” This definition consists of the mandatory factors including interplay, person, data interface, action-reaction, and data-related intent.

Using *intent* as a primary feature, Gotz and Zhou identified an interaction taxonomy consisting of three top-level classes of actions: exploration actions, insight actions, and meta actions [79]. Based on semantics, exploration actions are divided into two sub-categories: data exploration actions, including filter, inspect, and query, and visual exploration actions, such as brush, merge, zoom, and pan. Similarly, insight actions cover visual insight actions (annotate, bookmark) and knowledge insight actions (create, modify, remove). Meta actions help distill the units of user activity that should constitute an action, including delete, edit, redo, revisit, and undo. To ease the process of manually specifying visualization techniques, mappings, and parameters, Saket et al. proposed an interaction paradigm for visual data exploration, named *Visualization by Demonstration* [187]. A system that adopts this paradigm will enable users to provide corresponding visual demonstrations of incremental changes to the visual representation. Interactivity can be leveraged not only for visualization purposes but also for the preprocessing stage. Bernard et al. [22] introduced a visual-interactive approach to preprocess multivariate time series data, supporting data analysts to carry out six core analysis tasks related to preprocessing.

Interactivity is also a key component in visual analytics. Here, visual analytics is defined as the science that “combines automated analysis techniques with interactive visualizations for an effective understanding, reasoning and decision making on the basis of very large and complex data sets” [111]. For high-dimensional visual analytics, Wilkinson et al. [225] developed a method focusing on interactive exploration guided by pairwise views of point distributions. A more recent work by Nguyen et al. [163] adopted a user-centered approach in developing a framework for interactive visual user behavior analytics. This framework enables the analysis of collections of users and their multiple activity sessions within digital applications.

Recognizing the importance of interactivity in visual exploratory analysis, every visualization tool presented in this dissertation is equipped with interactive capabilities, supporting users to manipulate and explore data in real-time.

2.3 Time-series Data and Temporal Events

Temporal data is considered one of the seven basic data types most relevant for information visualization [197]. In this dissertation, a time-series dataset is defined as an ordered sequence of time–value pairs $\langle (t_0, v_0), (t_1, v_1), \dots, (t_n, v_n) \rangle$ [151]. In these datasets, time is one of the key attributes, as opposed to cases where the temporal attribute is a value rather than a key. These time-value pairs are often but not always spaced at uniform or equidistant time intervals. For example, measurement data is generated at equidistant temporal periods, but event-based data is not, and both are instances of time-series data [116]. A similar instance is event sequence, where the data records a series of discrete events in the time order of occurrence [87].

In 2011, Aigner, Miksch, Schumann and Tominski presented the book *Visualization of Time-Oriented Data* [5], introducing a systematic view and laying the structural foundation on how time-oriented data can be visualized to achieve the goal of understanding data and gaining valuable insights. Incorporating interactivity, McLachlan et al. introduced LiveRAC [134], a powerful visualization system designed to facilitate the analysis of large collections of system management time-series data. The system allows side-by-side visual comparison of arbitrary groupings of devices and parameters across multiple levels of detail. Gregory and Shneiderman presented an instrumental work on shape identification of temporal datasets [84]. The authors specified simple shapes commonly used to describe a behavior, including 1) spike and sink shapes, 2) rise and drop shapes, and 3) plateaus, valleys, and gaps. In a more complex case of high-dimensional time series, Dang et al. [56] presented nine Scagnostic measures for organizing multi-variate time series and guiding interactive exploration through high-dimensional data. These measures include density, skewness, shape, outliers, and texture.

Temporal events are determined at an atomic level as time-stamped data points or measurements [116, 16] that can occur at any time. A common scenario in time-series data involves temporal event sequence datasets, focusing on the captured events. In this context, events serve as inherent building blocks, and the dataset consists of one or more sequences comprising such events themselves [87]. However, when time series incorporates textual information, the concept of an event takes on a more complex structure. It may represent a unique pattern or irregular activity in terms of average signs across the temporal dimension [219], or an occurrence that leads to changes in the volume of text data characterized by a tuple of topics, time, people and location [67].

In the next chapter, we will look into a novel visualization technique combining the affordances of text and time-series visualization: WordStream.

CHAPTER 3

WORDSTREAM: INTERACTIVE VISUALIZATION FOR TOPIC EVOLUTION IN QUALITATIVE DATA

The contents of this chapter have been taken from previously published articles:

DANG, T., NGUYEN, H. N., AND PHAM, V. WordStream: Interactive Visualization for Topic Evolution. In *EuroVis 2019 - Short Papers (2019)*, J. Johansson, F. Sadlo, and G. E. Marai, Eds., The Eurographics Association. DOI: 10.2312/evs.20191178.

NGUYEN, H. N., DANG, T., AND BOWE, K. A. Wordstream Maker: A lightweight end-to-end visualization platform for qualitative time-series data. In *NLVIZ: Exploring Research Opportunities for Natural Language, Text, and Data Visualization (NLVIZ, Workshop jointly held with IEEE VIS) (2022)*.

Whether in the form of transcribed conversations, blog posts, or tweets, qualitative data provides a reader with rich insight into both the overarching trends and the diversity of human ideas expressed through text. Handling and analyzing large amounts of qualitative data, however, is difficult, often requiring multiple time-intensive perusals in order to identify patterns. This difficulty is multiplied by each additional question or time point present in a data set. A primary challenge then is creating visualizations that support the interpretation of qualitative text data by making it easier to identify and explore trends of interest.

This chapter introduces *WordStream* and *WordStream Maker*, as demonstrated in Figure 3.1. The former is an interactive visualization tool for the demonstration of topic evolution, and the latter is an end-to-end platform to enable the use of *WordStream* on raw text input. Our approach with *WordStream* utilizes the two popular techniques. Word clouds are designed to give an engaging visualization of text via font sizes and colors, while stacked graphs are a common method for visualizing topic evolution. In particular, *WordStream* emphasizes essential terms chronologically and spatially. To show the usefulness of *WordStream*, we demonstrate its applications on various data sets, including the *Huffington Post* and IEEE VIS publications.

Since it was first published in 2019, the *WordStream* technique has been applied in various scenarios, from social media content analysis to qualitative analysis in educational assessments to explore the evolution of text data over time. By combining the affordances of both text and visualizations, *WordStream* enables ease of information retrieval and processing of time-series text data. However, the data-wrangling necessary

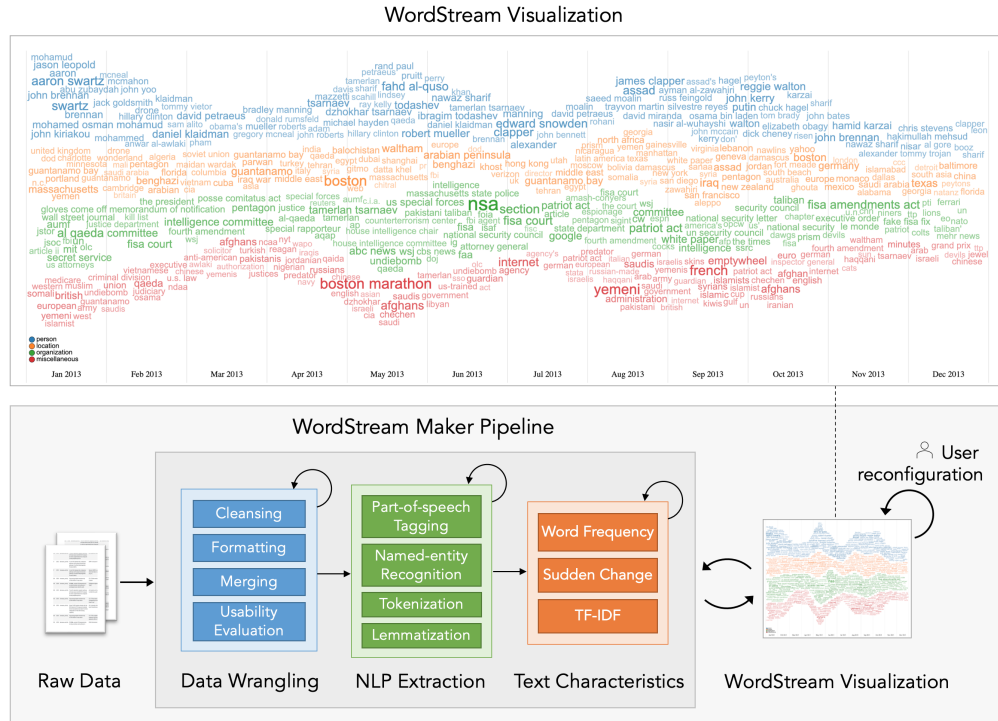


Figure 3.1: WordStream visualization and WordStream Maker pipeline.

to produce a WordStream remains a significant barrier for non-technical users without a programming background. In response, this chapter later presents *WordStream Maker*: an end-to-end platform with a pipeline that utilizes NLP to help non-technical users process raw text data and generate a customizable visualization. The processing of raw input data is embedded in the pipeline, and parameters are available for customization on demand. Lessons learned from integrating NLP into visualization and scaling to large data sets are discussed, along with use cases to demonstrate the usefulness of the platform.

We first published WordStream at EuroVis 2019 (co-authored with Tommy Dang and Vung Pham) [49] and extended it to support audiences without prior programming skill with WordStream Maker at NLVIZ: Exploring Research Opportunities for Natural Language, Text, and Data Visualization at IEEE VIS 2022 (co-authored with Tommy Dang and Kathleen A. Bowe) [159].

WordStream is an open-source project – with the visualization, demonstration video, and examples at datavisualizationlab.github.io/WordStream and its JavaScript library at github.com/huyen-nguyen/wordstream-library. WordStream Maker is also open source with the platform and gallery available at huyen-nguyen.github.io/maker.

3.1 Introduction

From everyday communication to analytical conversations, qualitative text data is around us in all shapes and forms. Documents, news, and transcriptions of audio and video recordings are examples of qualitative data; an area that has gained huge growth in recent years—social media data—is also considered qualitative data on a quantitative scale [69]. With the rapidly increasing amount of data over time, the question is: how can we best leverage the data that we already have to deal with on a daily basis into a more valuable asset?

The illustration of topic evolution has a long history. In 1931, *HistoMap* [198] was created by John Spark, showcasing the power of civilizations over four thousand years of world history. In more recent efforts, *ThemeRiver* [92] and, later, *StreamGraph* [28] expand the idea of the stacked graph to convey the evolution of topic [66]. Limited screen display and a large number of layers lead to the small area allocated for each term; therefore, the task of fitting terms into topic streams becomes more challenging. *Word clouds* [213] are designed for optimizing space usage [71]. However, temporal information has not been considered properly in this type of text presentation. The combination of word cloud and stacked graph models has been recently studied [203]. Nevertheless, there is still room for improvement and optimization, especially when the topic streams highly fluctuate.

Besides challenges in visual encodings, it is important to recognize that other facets of data transformation and processing present their own obstacles. Transforming typical raw data into usable data involves the process of cleansing, merging, formatting, extracting, and converting—generally known as ‘data wrangling’ [108] that amounts up to 80% of the development time and cost in data warehousing projects [58]. In qualitative data, this tedious task is coupled with the context-laden, conceptual characteristics of the data itself, making data processing increasingly challenging. With the advancement of natural language processing (NLP), the complex textual structure can be broken down and interpreted to assist humans in comprehension and communication. However, applying NLP to process text data remains challenging to non-technical users, especially when a large data set is involved. Hence, our primary aim is to build a visualization platform that helps bridge this gap and ease the process of extracting insights from temporal patterns in text data.

In response to the aforementioned challenges, we describe how WordStream and later, WordStream Maker, enable interactive qualitative data visualizations for exploration and analysis.

We first propose a synthesized approach to visualize information by means of word cloud within a stream layer to convey the evolution of topics over time in multiple categories. We implement an interactive text visualization prototype, named WordStream, to represent the evolution of topics to convey both spatial and temporal information. WordStream combines the advantages of Wordle [213] and stream graph [91] into a hybrid visualization to represent the topic streams, especially when they highly fluctuate. We evaluate the usefulness of the WordStream on various data sets (e.g., political blogs and other application domains). By preserving the terms in the visualization, WordStream enables both ease of information retrieval and information processing.

WordStream has been employed in various applications to explore the evolution of text data over time. The visualization was first introduced with topics from IEEE VIS publications, Huffington Post collection, along with other online blogs [49]. During the COVID-19 pandemic, the visualization is integrated into a tool called CovidStream to monitor the evolution of emotions associated with COVID-19 in Peru by Baca et al. [18]. The technique is also employed to analyze emerging topics on social media: assisting earthquake situational analysis [37, 157], cybersecurity [210] and natural resources [53]. Recent work shows the extensions of WordStream on educational assessments [161] and in the search for potential crowdfunding investors [231]. However, the use of WordStream library was confined to users and communities with proficiency in programming.

Second, as a natural extension of WordStream to expand the accessibility of the tool, we developed WordStream Maker to facilitate its use by a broader demographic, especially for users without prior knowledge of programming. WordStream Maker is an end-to-end platform to help 1) process raw time-series text data on the fly, 2) extract text characteristics with NLP, 3) generate the time-series visualization, and 4) aid users in customizing the representation. In *WordStream Maker*, the processing of raw input data is embedded in the pipeline, and parameters are available for customization on demand. The system is lightweight and runs entirely on the end user’s browser, with no server-side and database needed to reinforce data privacy.

3.2 Related Work

3.2.1 Word Cloud Models

Wordle [213] uses a randomized greedy algorithm to place words. It is greedy since it prioritizes the more frequent words. In addition, *Wordle* is aesthetically and visually appealing [71]. In the past years, there have been many efforts to optimize the *Wordle*

layout. *ManiWordle* [114] provides more flexible control over how to form the word layout with the interaction from the users. *Rolled-out Wordles* [202] makes use of Linear Sorting (*RWordle-L*) and Concentric Sorting (*RWordle-C*) to place the words more compactly and preserve the orthogonal ordering and topology. *WordlePlus* [104] extends the idea of *ManiWordle* to provide some further natural interaction supported for pen- and touch-enabled tablets while controlling the overall *Wordle* layout such as resizing, adding, and deleting elements. A recent work, called *EdWordle* [218], allows editing and preserving the word cloud layout. These works mostly focus on extending/optimizing the *Wordle* layout and discard the time element.

There are attempts to integrate temporal constraints into *Wordle*. *Parallel Tag Clouds* [39] utilize the parallel coordinates to represent time constraint. At each time step, terms are placed in alphabetical order or order of importance based on term frequency. This technique also has a feature implemented in *WordStream*, which displays a stream when a term is selected. However, *Parallel Tag Clouds* is not space-efficient and cannot show the topic or overall evolution across time.

The context-preserving word cloud [46] groups related words together so that the meaning of an individual word can also be inferred from its relationship with other words in the group. This method displays content evolution by using a set of word clouds and utilizes a separate significance trend chart to show the overall evolution. It cannot show the term evolution or topic evolution. It still needs to use a set of separated word clouds and a line graph. *Morphable Word Cloud* [38] specifies a sequence of shapes as boundaries of word clouds at each time step. These intermediate shapes of the sequence could also be automatically generated using interpolation from the specified first and the last shape. This is a series of separated word clouds at different time steps with shapes as boundaries and still contains spaces between shapes. In terms of utilizing word clouds, all these techniques use a separate set of word clouds at different time steps to show the evolution of terms. On the other hand, *WordStream* uses one *Wordle* across time to utilize all spaces between time steps and still preserve the topic and overall evolution using *StreamGraph*.

3.2.2 Time-series Visualizations

Stacked graph [91] has a baseline representing time constraints, and each layer serves as a topic of interest. The layers are stacked on one another starting from the baseline, and the change of the width of each layer represents the evolution over time. *The-meRiver* [92] is an optimization of the stacked graph, which provides a symmetric layout by setting the baseline at the center of the overall graph, which then helps to smooth

the transition across time. *StreamGraph* [28], an evolution of *ThemeRiver*, focuses on minimizing the wiggle per layer, which is the sum of squares of the slopes at each time point, to avoid legibility issue in the previous stacked graph and *ThemeRiver* versions, and also improve the aesthetic aspect of the overall graph. All these versions of the stacked graph have a common issue: the limitation in layer width when the number of layers increases [44]. With limited space for the stream layer, the process of locating terms of the topic within the stream layer [221] may encounter difficulties. Therefore, the stream layer typically contains only a few or no terms. This makes it difficult to view the evolution at a finer granularity [217]. Our WordStream places terms as close to its *time step* as possible by utilizing space sharing approach between adjacent time steps.

3.2.3 Combined Models

TIARA [125] utilized the combination of the word cloud and stacked graph to demonstrate visual representation from abstract and complex text summarization. The *TIARA* visualization includes keyword clouds embedded in the layers of a stacked graph, whose layers depict the different topics. However, *TIARA* only shows a few word clouds in the stream boxes where space is sufficient. Therefore, maximizing the space usage within the stream layer was not the priority of this technique. *TextFlow* [45] demonstrates evolution and relationships among topics and their critical events: birth, death, split, and merge. A word cloud at a particular timestamp can only be displayed on request. Our WordStream embeds terms directly into the stream layer. By using the space-sharing technique between consecutive word clouds and color-encoding for terms in the same layer, we present the global evolution of topic streams using their text elements.

3.2.4 Opportunities for Text Stream Visualization

In the state-of-the-art report on text data streams by Wanner et al. [219], word clouds were considered to be the only visualization technique based on textual data, while timeline and circular representations were more prevalent to visualize the evolution of the data over time. A more recent work by Ottley et al. [170] demonstrated that effective representations should take advantage of the affordances of both text and visualization. From an interdisciplinary perspective, a recent work by Nguyen et al. [161], indicated a gap between data visualization and qualitative research in visualization literacy and opened discussions on the opportunities for greater integration of the two fields.

Several directions are proposed to incorporate NLP into visual analytics. A natural language interface (NLI) in interactive visualization systems can offer graphical answers to address ambiguous questions from users [96, 192]. Narechania et al. [153] introduced

NL4DV, a robust Python package assisting visualization developers in creating new visualization NLIs and integrating natural language input into their systems without prior knowledge of NLP. While training and task framing remain challenges due to the characteristics of natural language input [200], by applying automated text mining methods in extracting and discovering knowledge in unstructured data sources [20], there are many open opportunities in the intersection of natural language and visualization.

3.3 Methodology

In this section, we describe the design decisions of and how to compute WordStream, the reasoning behind WordStream Maker platform, and its architecture as an end-to-end pipeline.

3.3.1 WordStream Design Decisions

This section presents the goals for designing WordStream and decisions made during the implementation of WordStream to meet these goals. The aim of WordStream is to communicate the global patterns of the text corpus across time. The design goals are largely drawn from the related work reviewed in Section 3.2.

- **G1.** Display the evolution of stream topics [45].
- **G2.** Emphasize important terms in their corresponding topic layers at their corresponding time steps [177].
- **G3.** Maximize the space usage and place as many terms within each topic stream layer as possible [72, 217].

The following decisions were made during the implementation of WordStream:

- **D1:** Apply *StreamGraph* to represent the topic evolution (**G1**). The time direction is from left to right [55, 56] while the height of each stream layer at a time step is relative to the total term frequencies [126].
- **D2:** Utilize a word cloud algorithm to display terms from each topic within its corresponding stream layer. This helps to satisfy **G2** [213, 71].
- **D3:** Place each term into its corresponding horizontal position according to its timestamp [39]. However, this constraint is loosened to *as close as possible to its timestamp*, so some terms may be placed outside their time boxes to meet **G3**.

- **D4:** Arrange texts with regard to their stream flows. Each word may have its own orientation (not always horizontal) to form the stream flows (**G1**).

Figure 3.2 shows a schematic overview of our approach. Topics are color-coded by using *displaCy Named Entity Visualizer* [147].

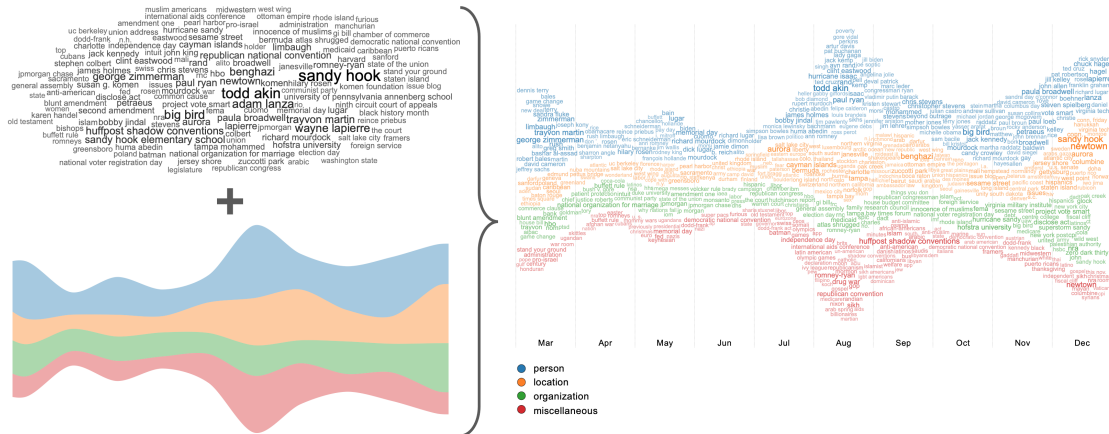


Figure 3.2: A standard word cloud and a stream graph are synthesized into a single snapshot (on the right).

On the control panel, users can customize the WordStream layout by adjusting the visual settings, such as the font scale, the maximum number of words in each box, and the dimensions of the overall layout. Regarding term orientation, we provide the following two options.

- **Flow:** The orientation of the words corresponds to its stream orientation at the time step that they belong to.
- **Angle variance:** The angle of rotation of the words varies within a fixed range with regard to the medium line of the stream flow.

3.3.2 WordStream: Computing The Visualization

The steps for computing WordStream visualization are demonstrated in Figure 3.3, including preprocessing data, building boxes, placing terms, and drawing.

Preprocessing data: The input text documents are preprocessed into entities and further classified into different categories [147]. In many cases, the term frequency might not convey user interests [57]. For example, the term “Obama” repeated numerous times in political blogs and news might not draw a lot of attention or interest [190]. To focus on the more significant terms, we use the *sudden attention* measure, referring to a

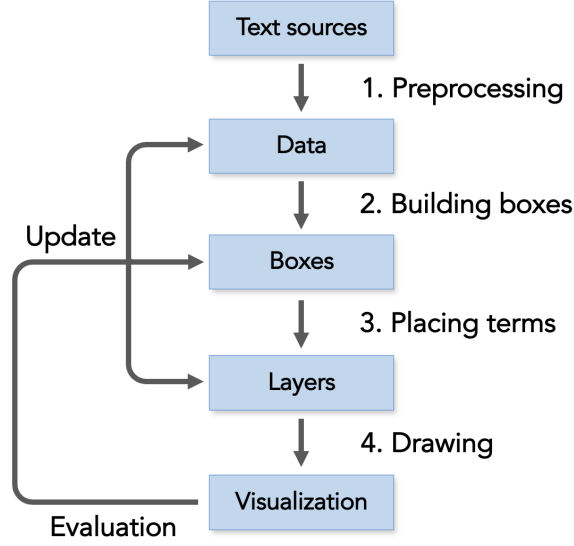


Figure 3.3: The main components of our WordStream visualization: Preprocessing data, building boxes, placing terms, and drawing.

sharp increase in frequency [52]. Let F_1, F_2, \dots, F_n be the frequency of an entity at n different time points. The sudden attention series (S_1, S_2, \dots, S_n) : $S_t = \frac{(F_t+1)}{(F_{t-1}+1)}$.

Building boxes: As for **D1** and **D3**, our approach places terms inside their corresponding stream and close to their time steps. Invisible boxes are created for this purpose, as depicted in Figure 3.4. WordStream scans along a spiral pattern centered at the box to find the first available space to place terms.

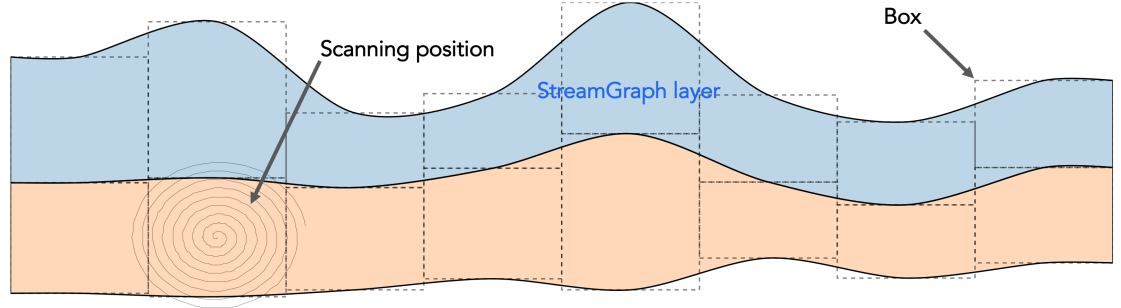


Figure 3.4: Building boxes and placing terms into stream layers.

Placing terms: *Mask-based*, *pixel-perfect* collision detection algorithm is used to detect collisions. The *Mask-based* algorithm uses a board to represent the stream layer with all the terms that have been placed on the board at the checking time. This board is used to check for collision against a new term. In the *pixel-perfect* approach, the terms and the board are represented in terms of pixels. To reduce the memory space,

only the red component (instead of all red, green, blue, and alpha components) is used to represent a pixel; this data is stored into a variable called *sprite* of the board or the term. The *sprite* value of a pixel i is computed from the pixel data using the following formula: $sprite[i] = pixels[i \ll 2]$. The collision detection checks the positions of all pixels on the sprite of the term and the corresponding positions on the board. Similarly, the placement of the new term onto the current board adds these values from the sprite of the term to the corresponding positions of the sprite of the board. As depicted in Figure 3.4, this spiral starts at the center of each box as calculated for its corresponding time step, and its maximum deviation from the center (dx , dy) is smaller than the diagonal of the box.

Drawing: The filtered terms are sequentially located to formulate the stream layers. Notice that the layers can be ordered vertical for quantitative data, such as in increasing order of security levels or user ratings. In addition, interactive features are supported to highlight individual term evolution (see Figure 3.5).

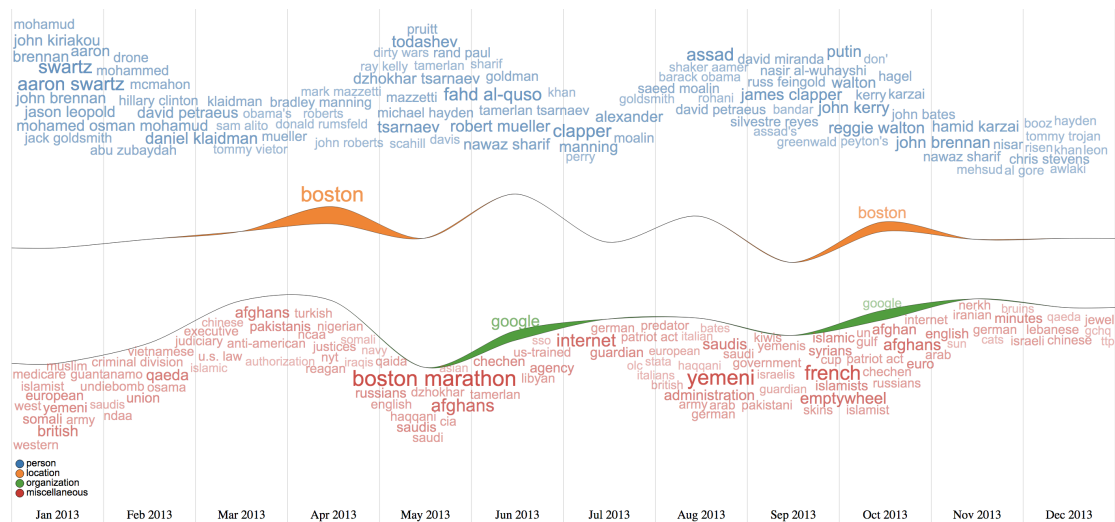


Figure 3.5: Term selection in the WordStream layout: *boston* (location) and *google* (organization).

3.3.3 WordStream Library

WordStream has been packaged as an open-source library in JavaScript at github.com/huyen-nguyen/wordstream-library. This section will show an example of how to use the library to make WordStream visualization and the data input format.

Figure 3.6 shows a simple HTML code snippet of creating WordStream, using embedded JavaScript inside HTML. Two JavaScript libraries need to be loaded:

WordStream library itself (line 6) and D3.js [26] for manipulating Document Object Model (DOM) elements (line 8). In this example, we use the minified version of WordStream to improve performance. The user specifies the SVG element in the DOM (line 14-16), then define adjustable parameters for configurations for WordStream (line 19-26), then load and visualize data in the JSON format (line 29-31).

```

1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <title>WordStream</title>
5   <!-- import WordStream library, minified version -->
6   <script src="js/wordstream.min.js"></script>
7   <!-- import D3.js -->
8   <script src="js/d3.v4.min.js"></script>
9 </head>
10 <body>
11 <script> <!-- embed JavaScript script within HTML -->
12
13   // set up blank canvas
14   let svg = d3.select("body").append("svg")
15     .attr("width", 1000)
16     .attr("height", 800);
17
18   // input configurations
19   let config = {
20     topWord: 20,           // top words to be selected
21     minFont: 10,          // minimum font size on display
22     maxFont: 30,          // maximum font size on display
23     tickFont: 12,         // font size of tick
24     legendFont: 12,       // font size of legend
25     curve: d3.curveMonotoneX // type of stream curve
26   };
27
28   // load data
29   d3.json("data/data.json", function (error, data) {
30     wordstream(svg, data, config) // visualize data
31   })
32
33 </script>
34 </body>
35 </html>

```

Figure 3.6: An example of how to import WordStream library to visualize data.

Figure 3.7 shows the simplest form of input data format as a JSON file. The default properties are defined as `date` (line 3) for timestamp, `words` (line 4) expecting the values as an object, where each property represents a category, `person` (line 5) is an example category. Each term entry consists of properties `text`, `frequency`, `topic`, `id`.

```

1  [
2    {
3      "date": "Jan 2013",
4      "words": {
5        "person": [
6          {
7            "text": "swartz",
8            "frequency": 7,
9            "topic": "person",
10           "id": "swartz_person_10"
11         },
12         {
13           "text": "aaron swartz",
14           "frequency": 7,
15           "topic": "person",
16           "id": "aaron_swartz_person_10"
17         },
18         {
19           "text": "john brennan",
20           "frequency": 15,
21           "topic": "person",
22           "id": "john_brennan_person_10"
23         },
24         {
25           "text": "brennan",
26           "frequency": 9,
27           "topic": "person",
28           "id": "brennan_person_10"
29         },
30         {
31           "text": "mohamed osman mohamud",
32           "frequency": 4,
33           "topic": "person",
34           "id": "mohamed_osman_mohamud_person_10"
35         },
36         ...

```

Figure 3.7: A snippet of the input JSON file.

The result of the above program 3.6 is shown in Figure 3.8. Although the visualization creation is simple and straightforward for developers or users with programming skills, it still requires a certain level of technical background to make use of the library. That is why we create WordStream Maker, to reduce the gap of prior knowledge on programming and establish an open platform for a wider audience. We discuss this topic in the following section.

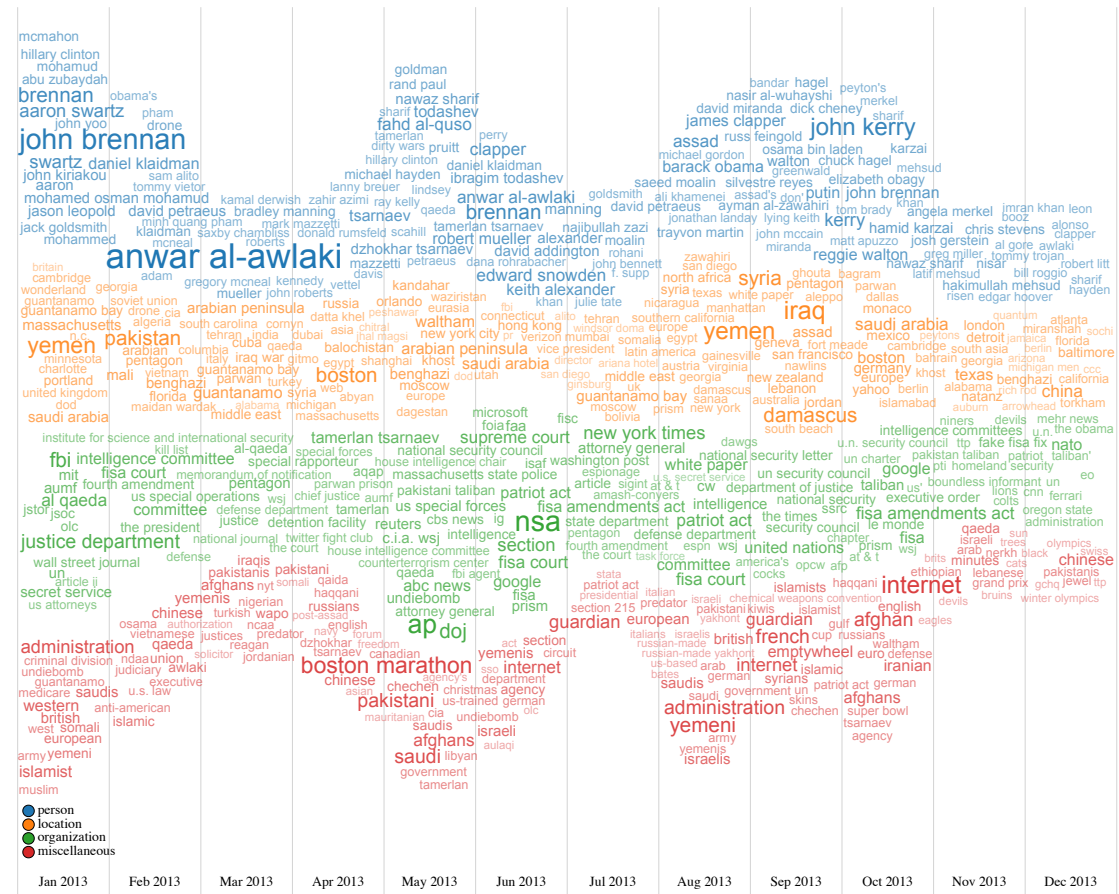


Figure 3.8: The resulting WordStream visualization from the above program and data in Figures 3.6 and 3.7.

3.3.4 WordStream Maker Platform

WordStream Maker is developed as a natural extension of WordStream to ease the creation process of the visualization by providing a platform allowing a straightforward pipeline from raw data to final visualization. WordStream Maker platform is inspired by several interesting existing works. First is the concept of the reusable chart, proposed by Mike Bostock [25], which provides reconfiguration of a visualization model

for customization flexibility depending on the user’s needs. Following this concept, RAWGraphs by Mauri et al. [131] introduces a visualization platform to create open outputs, opening many possibilities for encoding data dimensions on uncommon visual models. For static word clouds, the Word Cloud Generator developed by Jason Davies [59] offers a straightforward way to generate such clouds efficiently. However, there is still a lack of platforms for text data and qualitative data in general, especially text data coupled with the time element.

Given the opportunity to enhance the accessibility of the tool and bridge the gap between data visualization and qualitative research, we developed an end-to-end platform WordStream Maker to facilitate its use by a broader demographic, especially for individuals without prior programming knowledge.

WordStream Maker is an open source web application¹ based on mainly two libraries: D3.js [26] for the creation of WordStream visualization and Compromise [112] library for NLP engine.

3.3.5 WordStream Maker Architecture

The overall architecture of WordStream Maker platform is presented in Figure 3.1. First, the primary input to the platform is a raw time-series text file. Input data then goes through the main component of *WordStream Maker*, consisting of three sub-components: data wrangling, NLP extraction, and text characteristics computing. As the data source is heterogeneous, data cleansing is crucial to ensure the data is credible and usable. There are many cases where more than one row shares the same timestamp; hence merging data is the next important step.

The next sub-component is the NLP engine. This module provides Part-of-speech (POS) tagging, e.g., labeling whether a word is a noun, a verb, or other parts of speech such as preposition or adverb. POS tagging not only helps to extract the desired category but also makes it easier to remove stop words, which are commonly used and often bear little meaning. These stop words are often determiners, conjunctions, or prepositions; hence, we can skip these categories by using POS tagging. Diving in the Named-entity Recognition (NER) within the Noun category, three popular categories are chosen: Places, Person, and Organization. The application of these NLP modules in visualization will be discussed in Section 3.4.5. NLP engine also provides lemmatization, helping to trace the root form of any given word. This function is very useful in WordStream visualization; for example, the verb “study” in the present tense (study/studying) or past tense (studied) conveys the same topic; hence should be represented in

¹<https://github.com/huyen-nguyen/maker>

the same root word “study”.

The last sub-component is text characteristics computing, which serves directly to the visualization. The importance of a term can be represented by frequency, as in Wordle and other word cloud formations. In *WordStream Maker*, we use two other characteristics, namely sudden change and term frequency-inverse document frequency (TF-IDF). While TF-IDF is a statistical measure that evaluates the relevance of a word to a document in a collection of documents [180], sudden change is calculated based on sharp changes in frequency. Let F_1, F_2, \dots, F_n be the frequency of a word at n different time points. The sudden change series (S_1, S_2, \dots, S_n) : $S_t = \frac{(F_t+1)}{(F_{t-1}+1)}$.

3.4 Results and Discussion

3.4.1 WordStream Case Study: Exploring IEEE VIS Author Contributions

The *IEEE Visualization Publications* data set [101] contains 2,867 entries with attributes such as *Conference* (*InfoVis*, *VAST*, and *SciVis*), *Year* (from 1990 to 2016), *Paper Title*, *Link*, and *Author names*. This data set is particularly useful for finding appropriate authors for reviewing paper/proposal submissions and exploring the contributions of researchers over time. Figure 3.9 presents popular IEEE VIS authors over a period of 10 years. From the top down, we show the different generated layouts by combining the two options: **Flow** and **Angle variance**. As depicted, the top panel is the most *compact* layout as it can fit more author names than the others. This confirms our observation in the previous section. Moreover, the top panel also achieves the best readability as all terms are horizontal.

3.4.2 Informal User Study

We conducted informal user studies to gather qualitative responses about WordStream from two experts, one researcher in political science and one professor in data science. The study began with a brief description of WordStream to familiarize them with the usage of the analytic tool. We also adapted the implementation of *TimeArcs* [57] for the same political blogs as a reference. Then the experts were free to explore the visual interfaces for a specific task: *What are the top political events in the past ten years?* Both of them agreed that, in comparison to *TimeArcs*, the WordStream is useful to convey the global trend and can be applied to visualize emerging topics in various domains.

For the overall presentation, both of the users commented that they can quickly understand the idea of the layout. The data science professor stated that he had known word cloud before, and WordStream “allows you to do the longitudinal analysis easily.”



Figure 3.9: Popular IEEE VIS authors over 10 years from 2007 to 2016: author names are colored by their first publication venue.

He chose a term and scrolled through the entire timeline to see the fluctuation of its occurrences as depicted in Figure 3.5. Hence, the visualization is helpful in an exploratory analysis. However, he commented that the layout might be cluttered when the number of layers increases to about ten. On the other hand, the political expert at first found it “intimidating,” but shortly after the description, he found the interface easy to use. Furthermore, he found that brushing and linking are efficient for highlighting the temporal patterns of terms, along with supporting content analysis.

Besides the positive feedback, the experts also mentioned some limitations of WordStream, in which the related words are not shown in clusters, unlike *TimeArcs*. One of them suggested that the relationships being drawn explicitly among terms would be more useful than the proximity of terms as in the current visualization.

3.4.3 WordStream Quantitative Evaluation

For each test data set, the combinations of two options (**flow** and **angle variance**) are evaluated on the *Compactness* metric as depicted in Figure 3.10. *Compactness* is defined by the area of all displayed words divided by the area of the stream [19], indicating the level of coverage over the stream layer. *Compactness* has the range from 0 to 1; a higher value means the words are closer to forming the full shape of the stream layers.

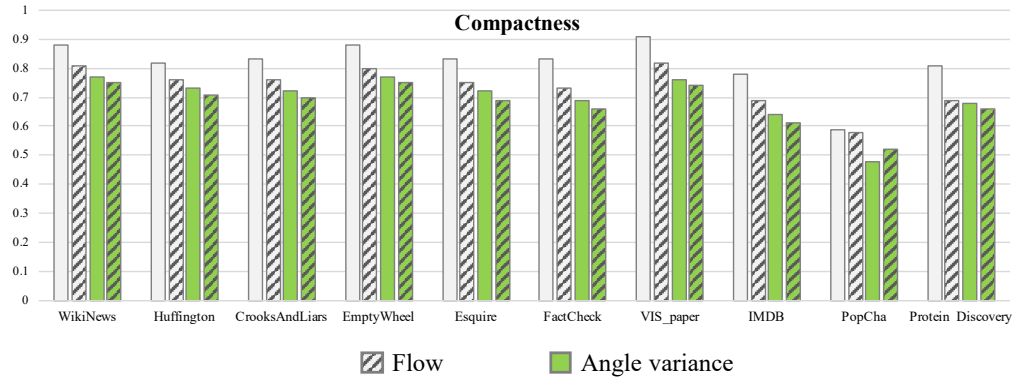


Figure 3.10: Comparisons of the Compactness measure for 10 test data sets (from left to right).

As seen in Figure 3.10, surprisingly, the combination that yields the best result is **disabling both features**. One example of this arrangement can be seen at the top panel of Figure 3.9. The conflicts in placing terms can be explained: If the variation is allowed, besides the conflicts from each sprite to one another, there are also conflicts from constraints in direction. In contrast, the combination of enabling both options produces the lowest *Compactness* scores on most test data sets.

WordStream is implemented using D3.js [26]. The demo video, online prototype, and more examples can be found at the GitHub page datavisualization-lab.github.io/WordStream.

3.4.4 WordStream Maker User Interface

The user interface of WordStream Maker is divided into three sections, as shown in Figure 3.11: (1) Data loading and preview, (2) WordStream visualization, and (3) Options for customization.

In the first section, users have the option of uploading a file containing tabular data in the tabular format of comma-separated values (CSV) or tab-separated values (TSV). For qualitative time-series data, the file must have one column representing the time element and one representing the body of text. There is also an option to load a sample data set for demonstration purposes. After the data is loaded, the window on the right will render a preview of input data as a table.

The second section is the WordStream visualization. At this point, the text data has been completely processed and parsed, and keywords have been extracted. The WordStream library will generate the visualization based on extracted data using a default configuration. The width of the stream presents the total frequency, and the font size of each word corresponds to its individual frequency (or other selected text characteristics).

The last section is for customizing the view: any adjustments will directly affect the WordStream visualization above. This section is divided into three groups: visualization-related, NLP-related, and text representation-related. In the first group, users can adjust the minimum and maximum font size in the WordStream, the number of words displayed in each stream for each time step, and the sizes (width and height) of the view. In the second group, the user can select to use POS tagging (default) or NER. POS tagging offers three categories: Noun, Verb, and Adjective, while NER corresponds to three main sub-categories: Person, Place, and Organization. In the last group, representation of text characteristics, the three options are frequency (default), sudden change, and TF-IDF.

3.4.4.1 Availability

WordStream Maker is available at huyen-nguyen.github.io/maker, and is an open source project. Interested readers can refer to the GitHub repository at github.com/huyen-nguyen/maker.

3.4.5 WordStream Maker Case Studies

The educational assessment data set [161] contains 63 entries with attributes such as *Course ID*, *Course Name*, *Prompt Text*, *Response Text*. This data set reflects the journal

entries where students give answers to weekly prompts made by instructors for class assessments. In this case, *WordStream Maker* is used for exploration without specific visual analytics tasks.

With POS (Part-of-speech) tagging and frequency option (Figure 3.12(a)), words are classified by Noun, Verb, and Adjective, with their font size indicating their occurrences in the text source. When we keep the frequency option and change POS tagging to NER (Named-entity Recognition) (Figure 3.12(b)), we can observe that: 1) the number of words falling into the new categorization greatly declines, and 2) there are only two prominent terms in the second view and they are of the same word “google”. This is because POS tagging provides more generic categorization, and the only organization that gains the most interest (based on frequency) is Google. When we keep the NER categorization and change to sudden change representation (Figure 3.12(c)), now the minor words in the previous view have the chance to shine, as there is a sharp increase in the mention of other organizations such as “microsoft”, “github”, “myspace”. This type of observation is useful in data exploration, especially in the early stage of the analysis process when the users may not be sure about what questions should be asked.

3.4.5.2 Data Exploration with Case Study on Protein Pathway Data

For different domains and different tasks, different ways of categorization should be applied to the qualitative data. Another use case is presented in Figure 3.13: Keywords from publications on protein pathways from 1996 to 2014. Summarizing from previous study [157, 53, 161] and *WordStream Maker*, we observe that: 1) NER categorization works best for data from social media or written form of response. 2) POS tagging can be employed for various domains; however, its generalization can be too broad to form a coherent story or specific insight.

3.4.5.3 Trade-offs

With the purpose of building a lightweight application, and the data processing part is meant to be on the fly, the selection on the NLP engine itself requires a modest-sized library. We go with *Compromise* due to the small size of its minified version (250kb) and the diverse NLP functionalities it provides. When it comes to tokenization—break a chunk of words into smaller units, there are two options. One is to break the sentence into individual words: a word is determined whenever we encounter the next whitespace. The other way is to break sentences into noun chunks, each containing one or more words; this way, we can preserve the meaning of the noun phrases. While the former

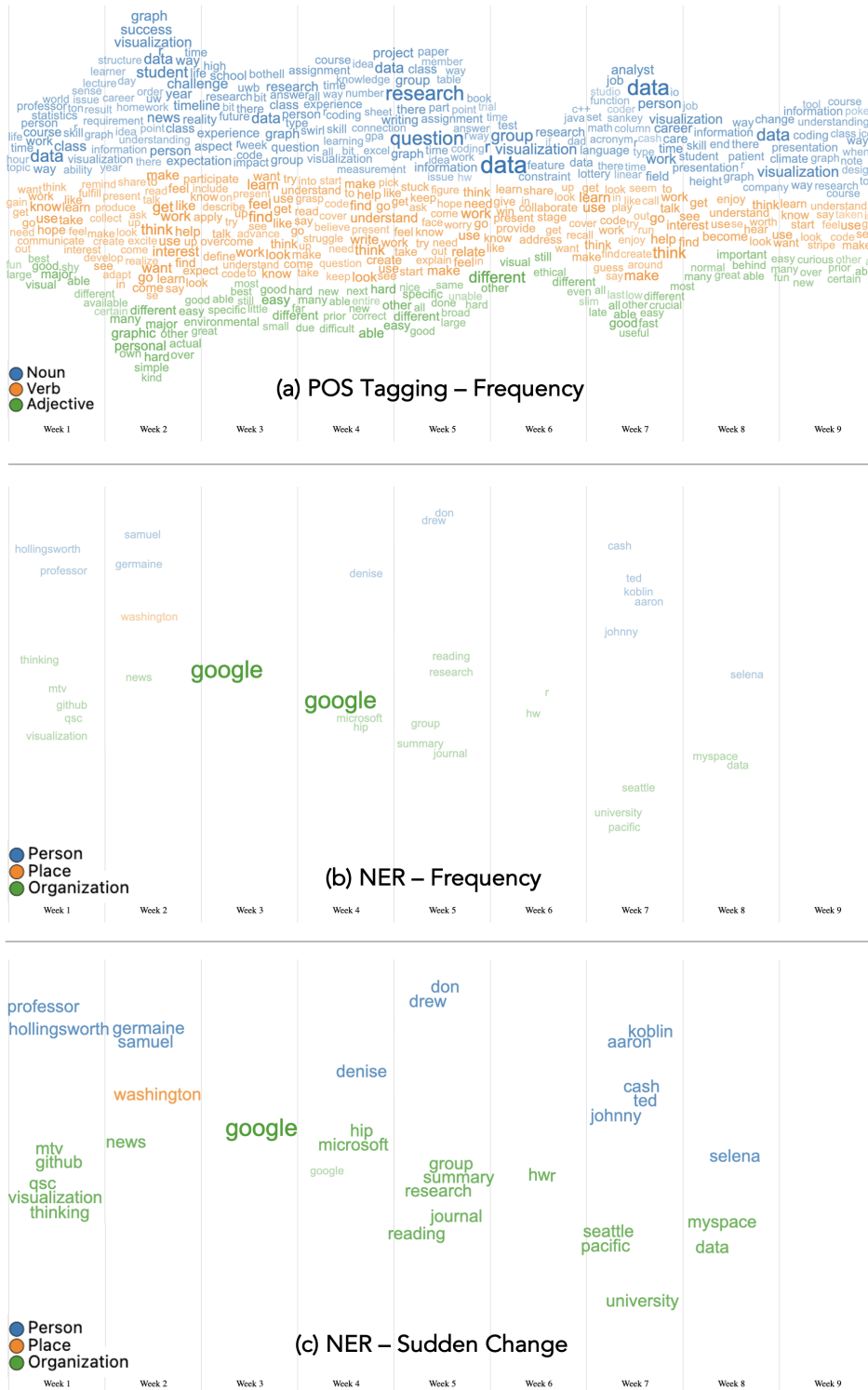


Figure 3.12: Journal data for education assessments throughout nine weeks. Different combinations of Part-of-speech (POS) Tagging, Named-entity Recognition (NER) in the NLP module, along with Frequency and Sudden Change in the text characteristics module.

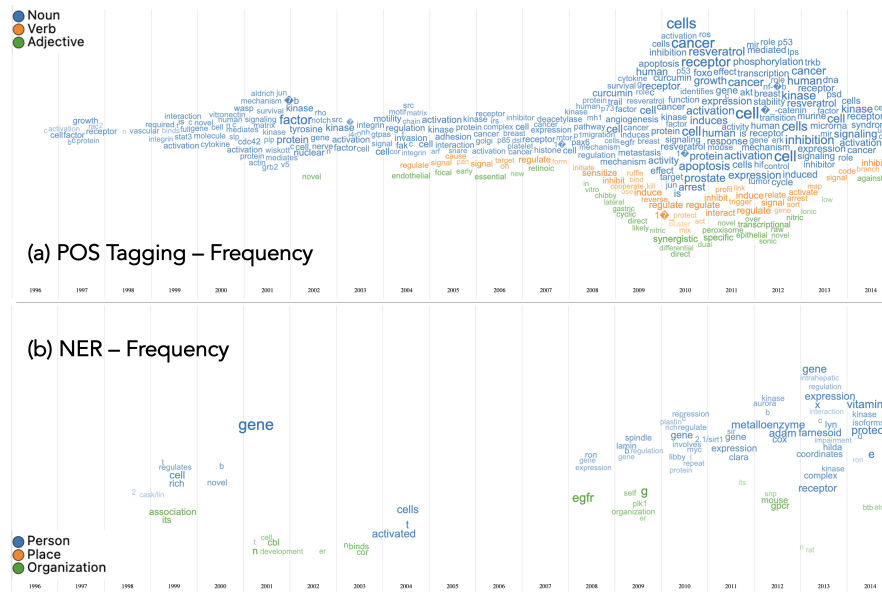


Figure 3.13: Keywords extracted from publications on protein pathways from 1996 to 2014. Different combinations of Part-of-speech (POS) Tagging and Named-entity Recognition (NER) with frequency.

approach is more straightforward, the second approach helps to maintain the semantics of text structure.

We run the experiment on a macOS Monterey 12.2.1 of 2.5GHz, and 16GB RAM. In runtime, for a text file of size 1.5MB containing more than 5000 rows, the first approach takes less than 3 seconds to load data, clean data, and run NLP extraction. However, the second approach takes 1 minute and 50 seconds, which is very time-consuming. With the aim of a lightweight platform, we choose simple tokenization at a small cost of semantics.

3.5 Conclusion

This chapter presents a hybrid text visualization approach along with a platform designed to facilitate its usage. WordStream aims to communicate the global trends of the underlying topic evolution while preserving the presentation-oriented criteria of the visualization solution. We demonstrate the applications on various data sets, showing that WordStream could quickly highlight important terms and could assist users in exploring term evolution at a finer granularity.

To foster accessibility to WordStream for a wider audience, we present *WordStream Maker*, a lightweight web-based platform to help non-technical users process raw text data and generate a customizable visualization of WordStream without programming

practice. The application is open source and meant to be expandable. Outlook for future work will focus on supporting more input types, interactive features, and applicability of WordStream Maker in a variety of use cases. To improve the algorithm of WordStream itself, further development will focus on algorithms to cluster the terms within and across topic streams.

CHAPTER 4

DEVELOPING QUALITATIVE DATA DASHBOARDS: CASE STUDIES WITH WORDSTREAM

The contents of this chapter have been taken from previously published articles:

NGUYEN, H. N., TRUJILLO, C. M., WEE, K., AND BOWE, K. A. Interactive Qualitative Data Visualization for Educational Assessment. In *The 12th International Conference on Advances in Information Technology* (2021), pp. 1–9. DOI: 10.1145/3468784.3469851.

NGUYEN, H. N. AND DANG, T. EQSA: Earthquake Situational Analytics from Social Media. In *2019 IEEE Conference on Visual Analytics Science and Technology (VAST)* (2019). DOI: 10.1109/VAST47406.2019.8986947. © 2019 IEEE.

Building upon the insights gained in the preceding chapter, we aim to advance further the technique of WordStream to enable rich and informative visualizations with qualitative data. This chapter explores the potential, constraints, and adaptability of WordStream in facilitating exploratory analysis through domain-specific dashboards. These dashboards cover different applications, ranging from social media content analysis to more specialized realms, such as analyzing educational assessments. Data visualization accelerates the communication of quantitative measures across many areas, including education. However, few visualization methods exist for qualitative data in educational fields that capture context-specific information and summarize trends for instructors. Informed by the Technology Acceptance Model, we used an informal user study to evaluate the perceived ease of use and usefulness of the technique for instructors using journal entries. Our evaluation found WordStream to be intuitive, clear, and easy-to-use to explore text entries, especially words of interest, but might be limited by focusing on word frequencies rather than underlying relationships among human ideas or other measures. Implications and challenges for bridging qualitative data with data visualization methods are discussed.

The first visualization dashboard and case study, Journal Data Dashboard, was built based on students' weekly journal entries collected as formative educational assessments from an undergraduate data visualization course and a statistics course. Journal Data Dashboard was published at The 12th International Conference on Advances in Information Technology, 2021 (co-authored with Caleb M. Trujillo, Kevin Wee, and

Kathleen A. Bowe) [161]. The second work, named EQSA, was developed using data from social media, supporting the exploration of community events during a series of earthquake-related incidents. EQSA was published at IEEE Conference on Visual Analytics Science and Technology (VAST), 2019 (co-authored with Tommy Dang) [157].

Journal Data Dashboard is available at edu-interactive-vis.netlify.app/. EQSA is an open-source project – with the visualization and demonstration video at datavisualizationlab.github.io/VAST2019mc3/.

4.1 Introduction

Qualitative data visualization aids in understanding and communication of qualitative data, where the data is descriptive and conceptual in nature. Presenting and interpreting qualitative data is a challenging task because of its nature, particularly being rich and context-laden, compared to numeric data that involves measurements and quantities. Qualitative data can be more difficult to analyze than quantitative data, as the data is not inherently objective or structured and therefore can be open to multiple interpretations [86].

When dealing with complex datasets, it has become a common practice to employ multiple coordinated views, which can be organized into visualization dashboards. For time-series data in particular, effective visualization should enable users to discern temporal patterns in unison with providing opportunities for deeper exploration through interactive features. The challenges for a qualitative visualization dashboard imposing context directly into data representation to observe the context-specific, longitudinal aspects of data and subsequently identify trends and diversity in the experiences are still underexplored. Additionally, the challenges of representing the complexity of qualitative data come with a lack of transparency in the analytical process [178].

A typical case of qualitative data can be found in education research. In an education setting, qualitative data takes many forms, including notes from classroom observations, a student’s response with comments from instructors, or a transcript from a teacher and parents meeting. Educational research can vary in scale, scope, purpose, and outcomes in using educational data. For instance, one project may include a multi-year ethnography of videotaped interactions in the classroom, or another project may gather student drawings as part of a single learning activity. More often, however, instructors gather qualitative data in the form of assessment data that is collected in the classroom as part of a way to gauge learning and to provide feedback in a formative way. These qualitative artifacts of learning, after being evaluated with a rubric, are typically reduced into single

dimensional values as scores, which provide quantitative measures of success but lose the nuanced knowledge structures and uniqueness of the students' responses. For example, a student's use of canonical and non-canonical languages in their responses reflects their learning progress on a topic, but these language features are difficult to represent using quantitative approaches or a simple word count. Here, we explore methods for visualizing qualitative data as a means to improve the way instructors gather insights from formative educational assessments.

In this chapter, we delve into the integration of WordStream in complex dashboards using two approaches: as a driving visualization or as a supporting panel in conjunction with other visualizations. Our methodology is grounded in two case studies focusing on qualitative data from educational journal entries and social media data. The first case study presented **Journal Data Dashboard** to discover the patterns and diversity of student ideas from journal entries in a context-specific approach. In our second case study, we built an Earthquake Situational Analytics (EQSA) dashboard that helped amplify the concerns of the community during a series of earthquake-related events. From the identified community needs, resource allocation and assistance were provided. The two case studies share the commonalities of analyzing time-series qualitative text data with the WordStream integration solution while simultaneously exploring two different domains with distinct domain-specific tasks. To conduct the evaluation for the qualitative visualization tool, we apply the Technology Acceptance Model (TAM) [60, 61] with perceived usefulness and perceived ease of use as a framework to measure users' perceptions of and intentions to use the interactive tool. Davis's TAM is the most widely utilized model of innovative technology acceptance and usage by users in terms of usefulness, attitude, and intention of users to use the technology [7]. The lessons learned from developing dashboards with WordStream, incorporating interactive features into larger dashboards, as well as our attempts to bridge the gap between data visualization and qualitative research, are discussed.

4.2 Related Work

From a review of 784 articles in three prominent qualitative research journals, Verdinelli and Scagnoli [212] found that only 23% of the articles included data displays. Among these displays, authors used various types: matrix (60%), network (12%), flow chart (9%), boxed text display, and Venn diagram. While some of these displays could be considered concept-driven rather than data-driven, the frequent use of matrices suggests the potential for growing data visualization in qualitative research. Similar findings

were reported by Henderson and Segal [93]. Despite the growing use of qualitative data visualization, the gap between data visualization and qualitative research is starkly evident in a recent data visualization literacy framework. Based on a review of over 50 years of work and 600 articles, Börner *et al.* [24] focus on four data types—nominal, ordinal, interval, and ratio data—to propose a plethora of frameworks and methods for displaying quantitative data, but under-explored rich qualitative data by grouping it as simply nominal.

While challenges for qualitative data visualizations are present, some notable examples are available that link traditions: quantitative ethnography’s use of epistemic network analysis [193], feminist ethnography’s use of geographic information systems in studies [113], and ethnographic arrays [2]. Thompson *et al.* [205] analyzed the way the structure of the learning environment affects the behavior of learners and vice versa, how that behavior has the potential to affect learning. The relationships between courses in a curriculum [105] and among concepts in a course [1, 160] with finer granularity are explored via multi-layered, multi-matrix visualization and node-link network and word cloud, as presented in the following.

Previous work has explored ways to visualize textual information. Wordle [213] generates a visual representation of words where the size of each word is proportional to its frequency within the text source. The underlying calculation of Wordle utilizes a randomized, greedy algorithm, where it prioritizes the more frequent words to place within the area. Alternatively, context-preserving word cloud [46] places a group of related words together so the meaning of an individual word could also be implied from its relationship to other words in its group. Building on these ideas, the Morphable Word Cloud [38] specifies a sequence of shapes as boundaries of word clouds at each time step. These intermediate shapes of the sequence could also be automatically generated using interpolation from the specified first and the last shape.

To encompass temporal constraint into the text visualization, WordStream [49] presents an integrated model, incorporating the ideas of the word cloud and stream graph to visualize topic evolution over time. The explicit context is characterized by words as topics and their distribution over time. The technique of WordStream is commonly used in visualizing a large corpus of text such as geo-tag messages from social media [37, 157] or social media discussions [53, 210] for further analysis on emerging topics. In this work, we leverage the use of the word cloud [213] and WordStream [49] to address the challenges of visualizing qualitative data embedded in the context of formative educational assessment setting and situational event characterizations from social media data.

4.3 Methodology

4.3.1 Case Studies: Descriptions

4.3.1.1 Case Study 1: Journal Data Dashboard

The formative educational assessment data is collected via students' weekly journals. This is followed by the process of NLP of the text data to extract part-of-speech tagging, topics, contexts, and a two-way mapping table. A web-based interactive tool is developed to visualize the assessment data with topic summary and topic evolution, providing linked views with interactive features. The interactive tool is evaluated using the Technology Acceptance Model via a user study regarding perceived usefulness and perceived ease of use. A schematic overview of the process of producing the interactive tool is demonstrated in Figure 4.1.

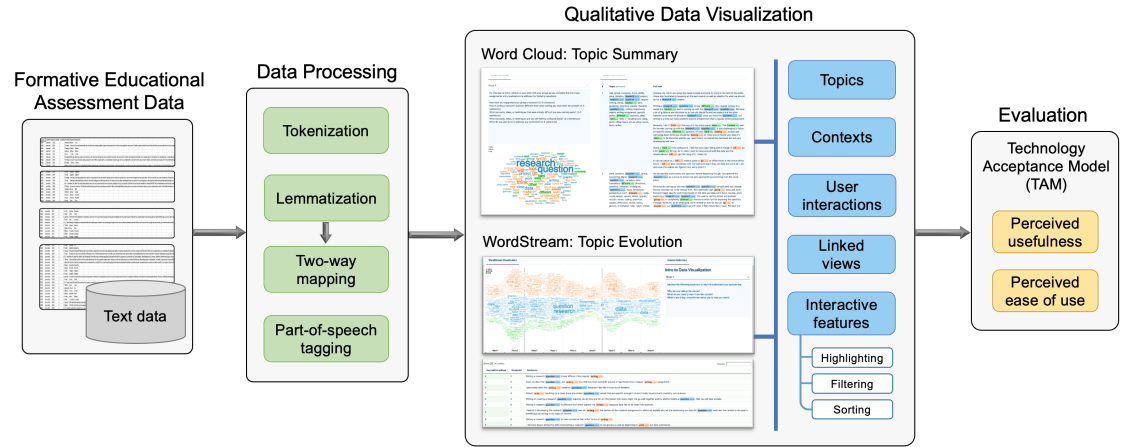


Figure 4.1: The schematic overview of the process to produce the interactive tool for visualizing formative educational assessment data. The visualization provides topics within contexts, user interactions with linked views and supports interactive features, such as highlighting, filtering and sorting.

Image description: Flowchart shows a schematic overview of four stages to research from left to right: (1) formative educational assessment data, (2) data processing, (3) qualitative data visualization, which shows the word cloud topic summary and word stream topic evolution and named features of the visualization, and (4) evaluation with the Technology Acceptance Model.

To explore novel methods for analysis and presentation as proof of concept, we collected complex qualitative data related to educational assessment data in the form of weekly journal entries. We collaborated with instructors of undergraduate research methods and data visualization courses in order to collect qualitative data as journal entries that include student reflections on the understanding of course content over time.

Participants wrote journals as part of their formative assessment in the 10-week course held in a computer laboratory on a regional community-serving campus. These assessments were graded for completion. Our data set includes 62 entries from 7 students (6 students from data visualization, one from statistics) collected in 9 weekly journal prompts, which contain 1 to 3 questions each. An example prompt (from week 3) used to elicit students' thoughts asks:

- What are two things you are learning that come easily and why do you feel that way (3 – 5 sentences)?
- What are two things you are learning that you find challenging and why do you feel that way (3 – 5 sentences)?
- What are two things you are learning that you want to work on (2 – 4 sentences)?

The responses made by students are useful (1) as a data set for visual exploration of qualitative data and (2) as a medium to track students' development of data visualization literacy. The research was performed under an Exempt status by the Institutional Review Board.

4.3.1.2 Case Study 2: Earthquake Situational Analytics

Recent years have witnessed an increasing use of social media as a digital channel for leveraging the voice of the people in the community as reactions and responses to natural disasters. The evolving online conversations contain valuable information that can be used to analyze the situations across the affected area, where survey data may fall short in terms of immediacy and the lack of direct reflection on a rapidly changing environment.

To explore such a data stream during a seismic event, this work presents an interactive visual analysis dashboard that integrates linked views of visualizations representing time, topics, emerging problems, and people within the community. The tool assists users in characterizing the conditions across the earthquake zone: determining occurring and post-earthquake events, identifying hot spots that demand emergency resources, and warranting facility re-allocations in the severely affected neighborhoods.

The data in this case study is a benchmark dataset provided by the IEEE VAST Challenge 2019 [41]. This dataset includes social media posts/messages capturing how people reacted to events associated with an earthquake, its aftermath and the subsequent impacts of those events on the community, based on a fictional city of St. Himark. The primary objective was to analyze text messages sent by citizens of St. Himark using

the social media application Y*INT. This analysis aimed to characterize the conditions within the city and formulate recommendations for the optimal allocation of resources. The main focus was on identifying relevant messages pertaining to the disaster and extracting the needs expressed by the citizens in order to effectively allocate resources accordingly.

The dataset consists of approximately 42,000 social media posts, spanning from April 06, 2020, to April 12, 2020, which has the following fields:

- time: date/time the message was posted
- location: neighborhood of St. Himark that the message was posted from)
- account: user handle of the person who posted the message
- message: text of the message itself

The earthquake analytics is implemented based on the workflow, including 1) characterizing the overall conditions by events that occurred and resources needed, then 2) discovering and exploring the story behind these situations on finer granularities of the data. In particular, the user utilizes the stream graph to detect the time points with a high volume of messages (for events or resources, or both), and a customizable time frame with a sliding window can adjust the scope of view. The interactive features in supporting panels characterize the situation with the chosen range, presenting detailed information and assisting users in gaining insights into the dataset.

4.3.2 Data Processing

To process the mass amount of text data collected, we apply NLP techniques. The processing stage is implemented with Python and SpaCy [95] library. Tokenization is used to extract words and phrases. Stop words- the most common words that do not add meaningful information to the text, such as “the”, “a”, “an”, and “in”, are eliminated afterward. Part-of-speech (POS) tagging is an important concept in NLP, where a label (such as noun, verb, or adjective) is assigned to each token to indicate the part of speech. We classify the words into three generic categories in our dataset: noun, verb, and adjective. Besides, we extract noun phrases in order to dive into the topics of interest further.

We created a two-way mapping of each token and its dictionary form for 1) Topic evolution visualization and 2) Original context being traced back upon a selected topic. The lemmatization technique is applied to map all words or phrases to their dictionary

form [34], where all the words describing the same root words are pulled together to the root. Compared to previous work [49, 213], lemmatization helps to bring the focus to the topic and remove the repetition of a topic disguised in many different forms of the root word.

With a large dataset of messages from a social media platform in the second case study, we first build a taxonomy for classifying these messages. For this particular earthquake dataset, we develop two main categories, which are “event” and “resource”. Sub-categories for “event” are earthquake, ground damage, flooding, aftershock, and fire; sub-categories for “resources” are: water, energy, medical, shelter, transportation, and food. Each sub-category contains a set of keywords to determine if a message belongs to it or not.

Besides, we have other options of *All* – total classified messages with all categories and *Other*, containing *Rumble* – the app for people to report about the disaster and *Other* posts – posts that do not have any of these above-mentioned keywords. We built the taxonomy with an assumption that one message can belong to more than one category, e.g., a message can indicate that there are needs for both water and food. The reasoning for this assumption comes from the possibility that priorities for resources may change over time; hence we need to take all information into account.

4.3.3 Design Decisions

This section presents the design goals to address the aforementioned challenges for a qualitative visualization tool and the decisions made in the implementation process to meet these goals. The resulting interactive dashboards are expected to serve these goals:

- **G1** Context-specific: Display the situational context along with visual representation.
- **G2** Data exploration upon multiple granularities in time-series data: topic-wise, time-wise, global and local trends.
- **G3** Details-on-demand: In addition to providing context (**G1**), the visualization should only present details upon interactions to avoid cluttered interfaces.

The following decisions were made during the implementation process of our tool:

- **D1** Associate the topic word to its context and highlight topics in a context upon user interaction (**G1, G3**)

- **D2** Utilize WordStream [49] algorithm to visualize time-series topic evolution (**G2**), representing the entry data along a horizontal timeline from left to right (**G3**).
- **D3** (Specifically for EQSA) Utilize word cloud [213] algorithm to visualize topics at a certain time point (**G2**)
- **D4** Enable multiple topic selections (**G2**) and emphasize significant topics to explore diversity and multiple views.

Besides these decisions, we explore two approaches to incorporate WordStream in complex dashboards: (1) as a driving visualization for Journal Data Dashboard and (2) as a supporting panel in conjunction with other visualizations with EQSA.

4.4 Results and Discussion

We implemented both dashboards as web-based applications using JavaScript and D3.js library [26]. The details on each interface are laid out below.

4.4.1 Case Study 1: Journal Data Dashboard

The interactive tool includes two main interfaces: The WordStream view and the Word Cloud view. The WordStream view demonstrates a time-series visual representation of the evolution of topics of interest, and the Word Cloud - summary view presents the focused topic within a specific week with details of students' answers. The demonstration and demo video of our tool can be found at edu-interactive-vis.netlify.app.

4.4.1.1 Summary View

The summary view contains three components, as depicted in Figure 4.2:

1. Journal weekly prompt selection
2. Word cloud for students' responses
3. Details in topics and responses

The journal weekly prompt section shows the journal questions of a selected week. The word cloud section summarizes the counts of words commonly used in students' responses to the journal questions. The font size of a word represents the word count.

The color of a word represents the part of speech (noun-blue, verb-orange, adjective-green.) Finally, details in the topics and responses section consist of individual student's responses to the journal questions shown in the tabular format, along with a set of topics (key phrases) generated from each response.

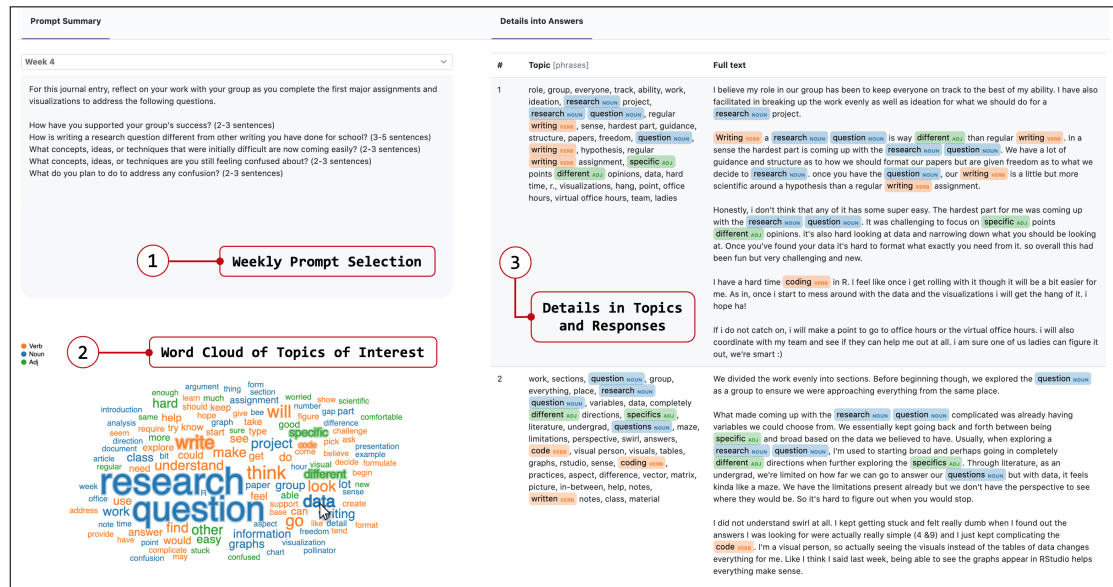


Figure 4.2: The topic summary view. The word selection from the word cloud includes: *research*, *question* (noun), *think*, *code* (verb), *different*, *specific* (adjective), resulting in the highlighted corresponding terms in the details. The word *data* is highlighted in the word cloud upon mouse-over interaction.

Image description: On the upper left, the weekly prompt given to students is shown with a drop-down menu to select different weeks. Below this, a word cloud displays topic words present in student responses, with different colors assigned to nouns, verbs, and adjectives. On the right, a table with two columns displays topic words present in a student's response beside the full text of the student's response. Each row is a different student. Words selected in the word cloud are highlighted in both columns according to their part of speech.

User Interactions

Selection Users can select a certain week or prompts that they want to inspect. They may scroll the text box below the selected week to read the full prompts.

Mouse-over Users can mouse-over a word in the word cloud to highlight it. For example, the word *data* (in category *noun*) in Figure 4.2 is moused-over and highlighted temporarily until the mouse moves away from the word.

Mouse-click To explore multiple selections, users can click on one or multiple words in the word cloud to highlight the words in the word cloud (2), individual student re-

sponses, and topics (3). Note that words with the same stem in the same part of speech (e.g., work and working as verbs) will be highlighted simultaneously. For example, the words “*question*” and “*specific*” in Figure 4.2 were clicked and highlighted. Besides, users can click a word again to deselect it.

Exploring the Topics and Opinions By multiple topic selections, we can observe and shift our focus onto the portion of opinions that the students express in their responses. With the selection of *writing*, *research*, *question*, *different*, and *specific*, the instructor can follow the thought of “Writing a research question is way different than regular writing. In a sense, the hardest part is coming up with the research question.”. Besides, the selected adjectives help orient the sentiment the instructor is looking for, “It was challenging to focus on specific points [of] different opinions.”

In the details in the topics and responses section, from a quick glance at the *Topic* column, we can summarize what the response is about and whether or not it aligns with the selected topics from the word cloud. In Figure 4.2, we can see that Student #1 are more concerned with the *writing* aspect than Student #2, via the three *writing* terms highlighted, compared to only one *written* term of Student #2. This observation is verified by inspecting the details of the responses.

4.4.1.2 Topic Evolution View

The WordStream view contain the following three components, as shown in Figure 4.3:

1. Journal prompt reference
2. WordStream view for topic evolution
3. Students’ responses.

The journal prompt reference section provides the background to the weekly journal questions, presenting the context for each week’s data shown in both the WordStream view and students’ response section. Next, the main view of the WordStream section presents the global patterns of the formative educational assessment data over time. The timeline is demonstrated horizontally from Week 1 to Week 9. The font size of each word represents its frequency in the corresponding time point it appears, while the color of the word indicates the category it belongs to. The frequency of a word can be used to represent its significance via font size, while another metric such as “sudden attention” can be applied when we want to refer to a sharp increase in frequency [49]. The text corpus is classified into three categories corresponding to three major parts

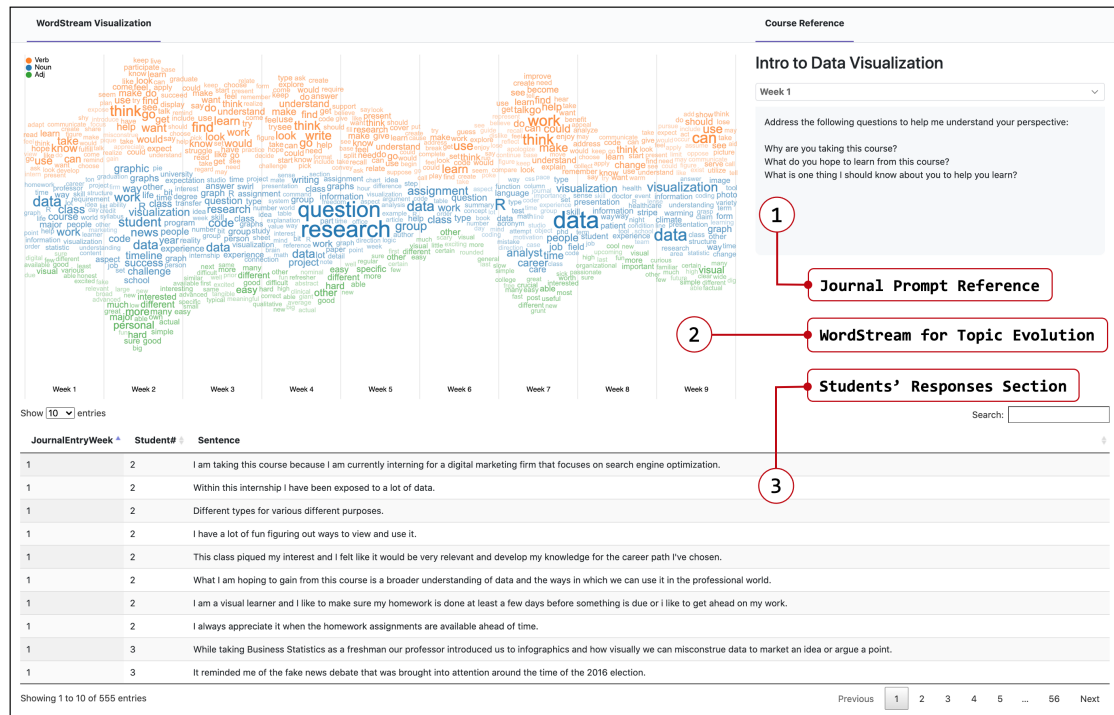


Figure 4.3: WordStream visualization: A time-series visual representation of the evolution of topics of interest. The visual components: (1) Journal prompt reference to weekly journal questions, (2) WordStream visualization for topic evolution, and (3) Students' responses section that helps explore into details.

Image description: The horizontal x-axis shows the weeks of the semester, from week 1 to 9. There are three groups or streams of words, colored according to part of speech (verb, noun, and adjective). The streams are of different thicknesses and vary in thickness over time. On the upper right, the weekly prompt given to students is shown with a drop-down menu to select different weeks. Across the bottom of the interface, there is a table with three columns with the headings 'Journal Entry Week,' 'Student #' and 'Sentence.' Each row corresponds to a sentence from one student for a particular week.

of speech, as indicated in Section 4.3.2: noun, verb, and adjective. The visualization is formulated as multiple streams, each for one category. The change of the width of each layer over time demonstrates the global patterns and represents temporal evolution. Finally, the students' responses section includes the details into students' data. Each record in the table is a separate sentence from the student's response that contains the selection word(s). The details of these visual components upon user interactions are presented in the following section.

User Interactions With user interactions, users can gain insights into the local pattern of a single topic.

Mouse-over On a single mouse-over, the appearances of that topic across the timeline are highlighted. For example, the word *easy* (in category *Adj*) in Figure 4.4 are emphasized along with its occurrences in other time points.

Mouse-click To explore further, the user can click on a word, then the words’ corresponding stream graph showing the changes in its frequency over time is displayed. For example, the words “*learn*” and “*data*” in Figure 4.4 are clicked, quickly demonstrate their temporal distributions.

Table interactive features The students’ response section contains the features to assist data reading:

- **Sorting by attribute:** Users can sort the table based on the week, student number (whose real ID was redacted due to privacy), or the sentence alphabetically.
- **Paging:** Users can adjust the entries shown on one display and therefore have multiple pages of students’ records. This is for keeping all visual components on one page.
- **Filtering:** The feature allows users to find all records that contain the input of the search box. For example, the word *visualization* is put in the search box, and the responses are filtered again only to keep the records containing *visualization*.

Exploring the Temporal Patterns The WordStream view allows the user to observe the global pattern (Figure 4.3) or the local trend (Figure 4.4). In Figure 4.3, Week 2 has the largest amount of responses, but the topics are smaller and more discrete, compared to a major topic such as *question* and *research* in Week 4 or *data* in Week 7. The blue stream in Figure 4.3 represents the changes in usage frequency of *data* from Week 1 to Week 9. The width of the *data* stream heightens at the last three weeks, demonstrating that the focus on data is emphasized towards the end of the course, along with the research report on data visualization. In the responses section, with the selection of *learn* from the stream, and filtering by the search box with *visualization*, we can see the reflections of students on learning data visualization, “I know data visualization is important, and I want to learn how to do it because I don’t understand just charts of data.”, or making inferences to apply into other domain: “Through data visualization, [...] this is a simple and easy way for people who want to learn about climate change and those who don’t believe climate change is a real problem to actually view the stats behind it.”

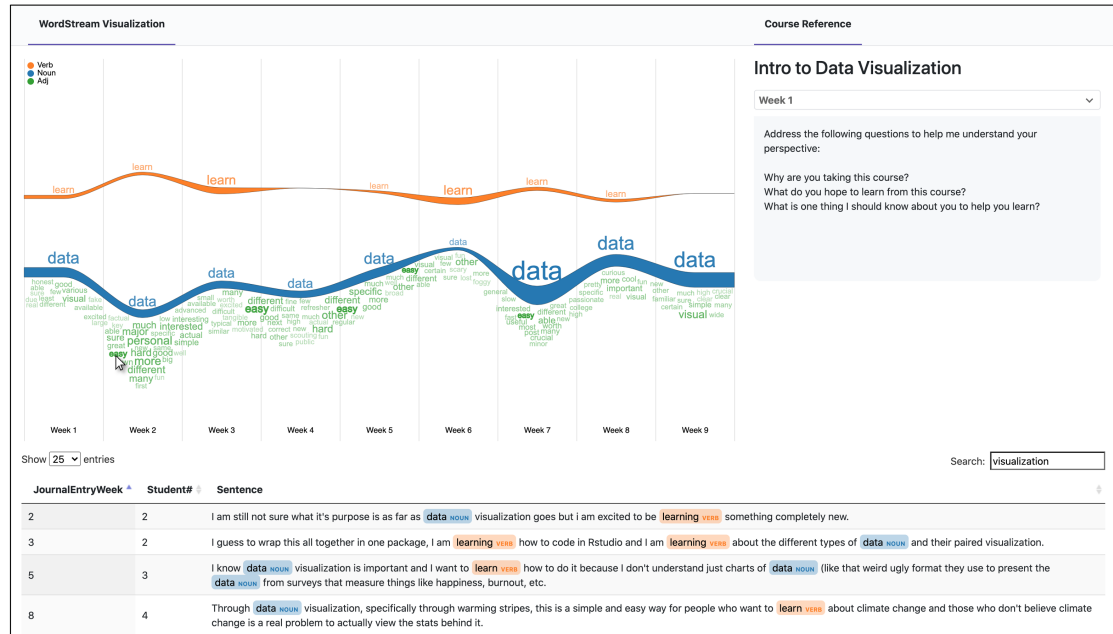


Figure 4.4: User interactions on WordStream view. There are several interactions are at play here. First, the words “learn” and “data” are clicked and the two corresponding streams representing the changes in their frequency are shown. The students’ responses are filtered to show only the records containing “learn” and “data”. Second, the word *easy* is mouse-overed, hence its occurrences in other time points are highlighted. Third, the search input is *visualization*, thus the responses are filtered again to only keep the records containing *visualization*.

Image description: On the upper left is the WordStream. The horizontal x axis shows week of the semester. The words ‘learn’ and ‘data’ have been selected from the verb and noun groupings or ‘streams’ of words. The verb and noun streams have collapsed to dark lines of varying thickness, showing the usage frequency for the selected words over time in the data. The adjective steam is still a group or stream of words. On the upper right, the weekly prompt given to students is shown with a drop-down menu to select different weeks. Across the bottom of the interface, the words selected in the WordStream are highlighted according to their part of speech in the sentences in the table with three columns (headings ‘Journal Entry Week’, ‘Student #’ and ‘Sentence’). Each row corresponds to a sentence from one student for a particular week.

Similar to the word cloud view in section 4.4.1.1, the selection of topics helps to limit the responses to certain topics of interest. While the WordStream view provides the temporal pattern clearly, there is a restraint on the number of topics that can be selected for further examination, compared to the multiple selection feature in the word cloud view. To compensate for this trade-off, the interactive tool is integrated with a search box, supporting the user to retrieve the information on demand.

4.4.2 Case Study 2: Earthquake Situational Analytics

4.4.2.1 Dashboard Interface

Figure 4.5 shows the main components of EQSA with coordinated views of multiple panels, including WordStream. Detail-on-demand is provided upon mouseover for all panels. EQSA is an open-source project, with the visualization, demonstration video and corresponding entry [158] at datavisualizationlab.github.io/VAST2019mc3.

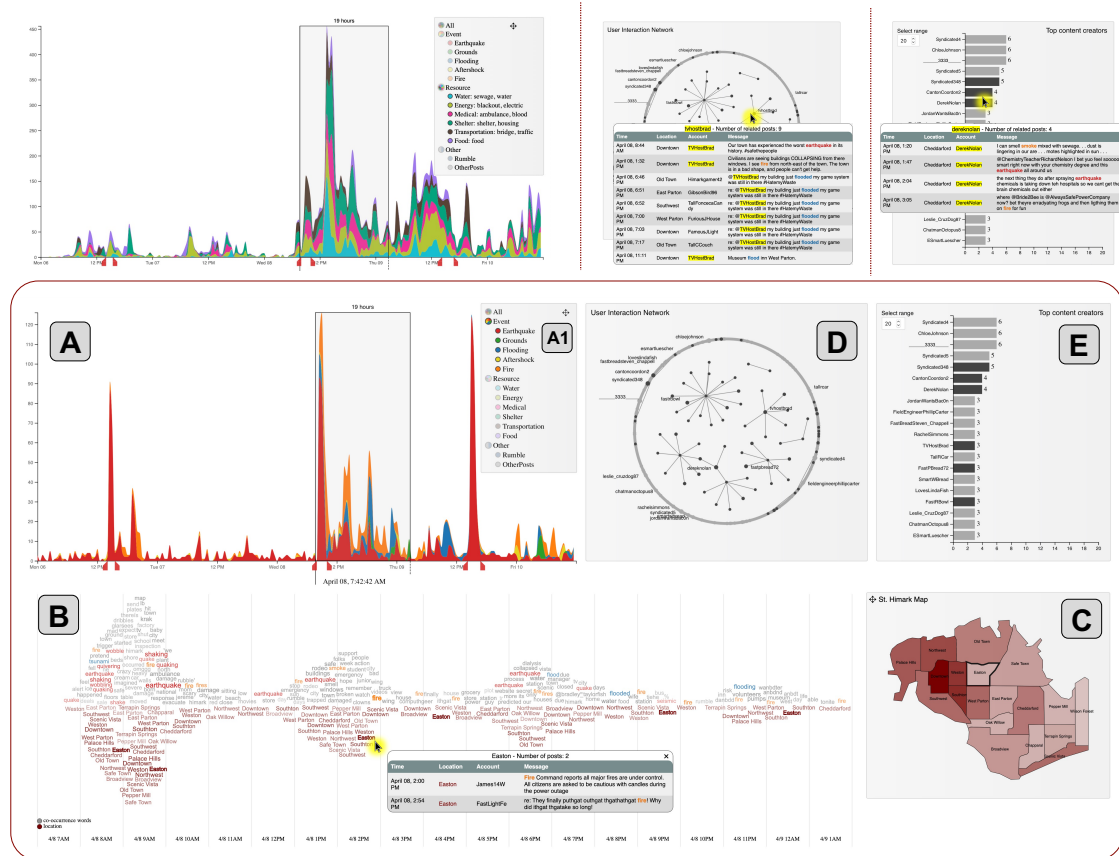


Figure 4.5: EQSA Earthquake Situational Analytics dashboard interface. The main control panel (A) is built as a stacked area chart, showing the volume of messages according to chosen categories from the selection panel (A1). For the chosen time frame and the chosen categories: Panel (B) presents topic evolution, panel (C) is a map in which each neighborhood's color indicates the number of posts, panel (D) is a network user interaction, and panel (E) is a chart for ranking content creators. Interactive features are provided to support the analysis, including a sliding window for time frame selection with *adjustable* window size, mouseover events providing detail-on-demand messages (at the black arrows). In panel (A), three high red peaks correspond to three times the earthquake strikes. Three panels at the top of this figure show the filtering option corresponding to “Resource” instead of “Event” as in the bordered panels below.

For time-series data, the stacked area chart is frequently utilized for visualizing changes of data over time for multiple categories. We utilize this chart type to represent the volume of classified messages and provide an overall view of the condition. To explore the detail, a sliding window on top of the area chart is provided for brushing and selecting the interested time frame. Besides the overall or global trending of terms, WordStream represents the local evolution of an individual topic. Regarding community, a network of users represents the existing clusters based on the relationships among users: the source is the sender and the target(s) are the users that are mentioned in the message. This strategy is used to detect who are the users that have multiple connections in the community; hence, they are likely to spread important messages. The location and ranking of users by amount of content are provided with a map and a horizontal bar chart.

The dashboard includes interactive components within a coordinated linked view. The main control panel (A) is built as a stacked area chart, showing the volume of messages according to chosen categories from the selection panel (A1). The vertical axis shows the number of posts, while the horizontal axis shows the timeline. A sliding window across the timeline has expandable width from 1 hour to 31 hours. For each change in the main panel (A) – whether it is adjusting time frame, timestamp or category, all other four panels (B to E) are updated according to (A).

Panel (B) provides a WordStream, demonstrating the evolution of topics extracted from the messages' content. The WordStream consists of two topics: the keywords within the content of messages and the location of the message. The thickness of the stream is proportional to the number of posts – the global trend. Users can also explore the local trend of an individual term and the detail of messages.

Panel (C) is a geographical map in which the color of each neighborhood indicates the number of posts. Users can use this map to highlight corresponding messages from a location in the WordStream and vice versa.

Panel (D) is a network of user interaction. The network demonstrates the connection between users, through the mentioned accounts in the content of messages. Via this network, we can spot which account has an essential role in the community.

Panel (E) is an account list for ranking content creators. This chart shows the accounts that create the largest number of posts. Users can see who writes many posts and their connection to the community; exploring these points can help detect irrelevant accounts.

4.4.2.2 Use cases

Earthquake and its aftermath impacting the community

From the main streamgraph view in Figure 4.6, the filtering option of “Earthquake” clearly shows three spikes indicating the earthquake strikes three times. We used these peaks to specify the timestamp earthquake strikes: (1) at around April 06, 2:31:34PM, (2) April 08, 8:34:18AM and (3) April 09, 3:03:20 PM. To validate these points, we explore the detail from corresponding messages. The WordStream supports highlighting the keywords in the corresponding categories. When mousing over terms, WordStream highlights the co-occurrences of words as well as related keywords from the messages. The locations of these messages are also emphasized in the lower “Location” stream and in the geolocation map. On April 6, from 2PM to 3PM, there were 36 messages about earthquake. At 2:33 PM on April 6, at Safe Town, an earthquake was recorded. This would be the timestamp with the highest accuracy with regard to the first strike.

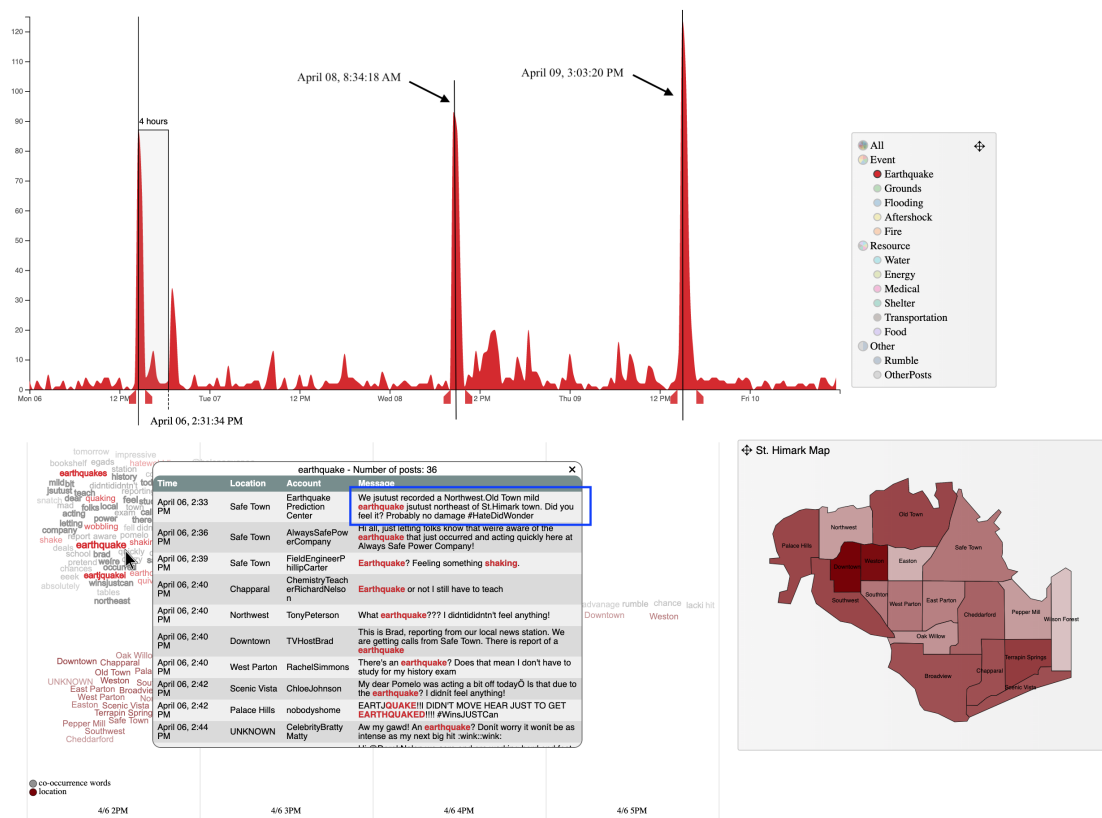


Figure 4.6: Three times the earthquake strikes (upper panel) and details for first strike (lower panel). Details-on-demand with WordStream shows seismic event messages.

Related events that occurred during and after the earthquake are included to characterize the condition as a whole. Besides the earthquake strikes, the city also suffered

from related events, including Floods, Ground damage, and Aftershock. Especially, there was a fire happening in a broad area.

Flooding In this section, the flood event is examined. In panel (b) in Figure 4.7, the keywords related to "Flood", such as "flooded", "flooding" are taken into account. From this approach, we specify that the neighborhoods that have been flooded includes: Old Town, East Parton, West Parton, Cheddarford, Northwest, Scenic Vista, Terrapin Springs.

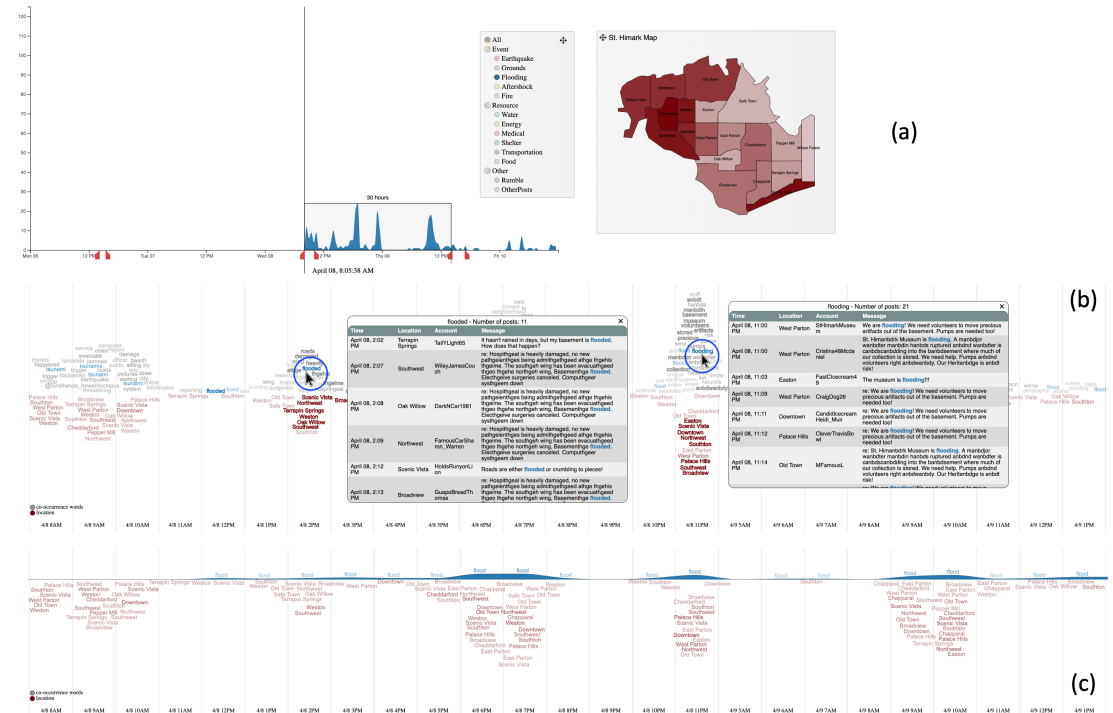


Figure 4.7: (a) Event of flooding; (b) Detail of events that happened in the city and their corresponding locations; (c) Trending of flooding event concerns over time.

Fire There is a fire in a broad area. According to Figure 4.8, the fire started at about 8:38 AM on April 8, and kept spreading to the area of Scenic Vista, Northwest and Oak Willow (severe), also: Old Town, Downtown (the fire Downtown spread for a long time), Broadview, Easton, Weston and Southton.

Key influencers in the community

Besides taking the messages from the keywords, we verify the events through the accounts with important roles in the community. We define these accounts by their interaction and influence on other accounts. By sliding the window along the timeline, we can spot who has multi connections to the community over time. From these accounts,



Figure 4.8: (a) Event of fire; (b) Detail of events that happened in the city and their corresponding locations; (c) Trending of fire event concerns over time.

we can determine what the community is experiencing since these accounts tend to raise their voice for the community. In Figure 4.9, Emergency Manager (left) and dereknolan (right), are two accounts that exhibit significant influence on the community.

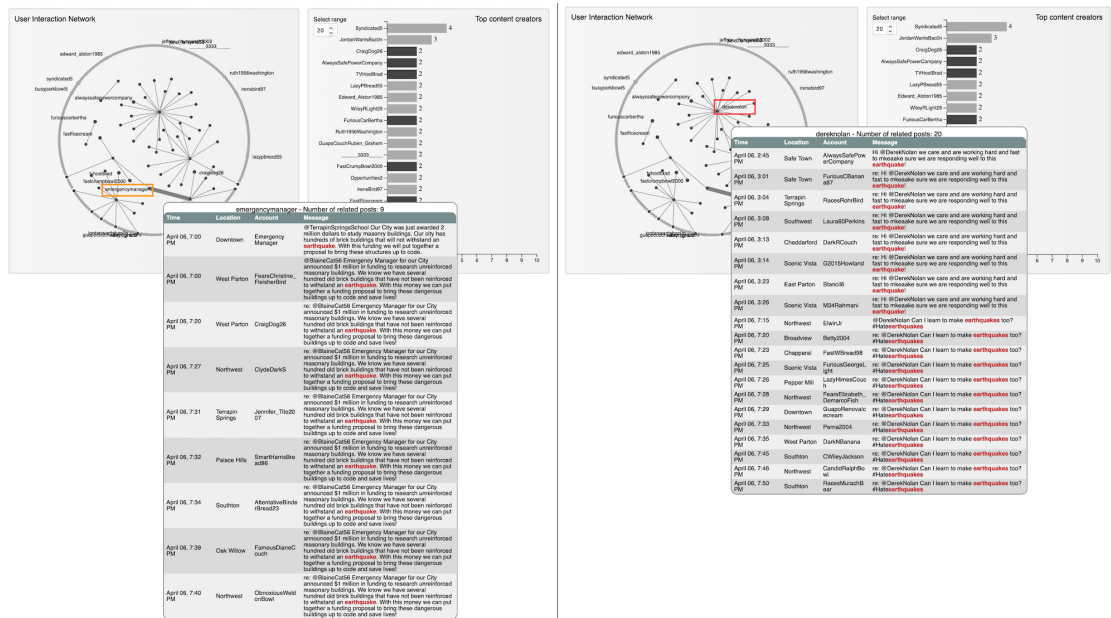


Figure 4.9: Emergency Manager (left) and dereknolan (right), two accounts that have influence in the community.

Timeline-based views and network view

Figure 4.10 shows the visualization with data from five hours after the third strike. The sliding window in the stacked area chart has a width of 6 hours. Inside the window, the brown stream - transportation has two major regions, one arises from right when the earthquake strikes and one towards the end of the window, showing that issues with transportation occurred within this time; we can come up with the observation: The problem occurred, and then it was addressed.

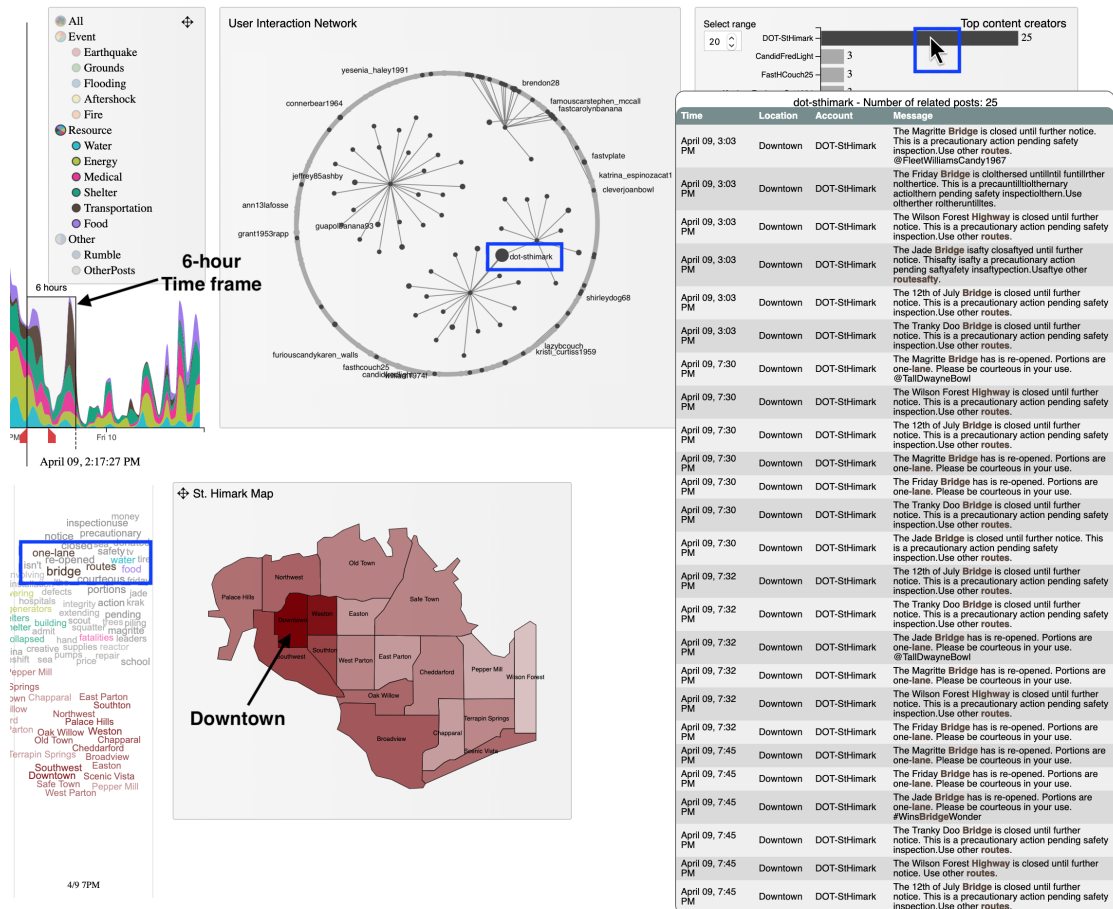


Figure 4.10: Five hours after the third strike, the main problem comes from “bridges” and “route” (terms emphasized in the WordStream), with multiple notifications from the Department of Transportation of St. Himark. © 2019 IEEE.

Three blue boxes are highlighted, from 1) The WordStream, 2) The network and 3) The ranking of content creators. In the WordStream, “bridge”, “one-lane” and “routes” are emphasized, supporting the observation from the main panel. Moving on to the network, the node of “dot-sthimark”, the DoT - Department of Transportation of St. Himark, is in large size - an important component of the community at this time. This

means that the community is connecting to the department for information. On the ranking bar chart, the department provides 25 notifications within 5 hours, a large number compared to the next ranked account (only 3). In the map, the neighborhood of Downtown is in the darkest shade - with the biggest number of posts, because the DoT is located in Downtown. By exploring the detailed messages from the department, we discovered that the bridges to get in and out of St.Himark are closed and then re-opened for a safety inspection. This confirmation helps validate the observation mentioned above. The complementary information provided from the timeline-based views (stack area chart, WordStream) and network view inspire this dissertation to explore a unified framework that utilizes both of them. This direction is later presented in Chapter 6 with an automatic event detection framework.

4.4.3 Informal User Study

This section presents an informal user study conducted to collect feedback about using the Journal Data Dashboard tool. Given the exploratory nature of this work and being served for a niche audience, a small number of instructors familiar with data visualization were invited to participate. We gathered qualitative responses from three experts: one professor in research methods who teaches data visualization; one professor in STEM; and one professor in computer science. Qualitative responses were collected in order to gain a more detailed understanding of each instructor's experience using the tool and to inform the future development of the visualization.

Each expert was provided with a brief tutorial of the tool, including written instructions and short animated clips to demonstrate different features and actions. The experts were asked to evaluate the tool as a way to visualize and explore student assessment data over time from the viewpoint of an instructor. To evaluate and measure the instructors' perceptions of and intentions to use the interactive tool, we applied the Technological Acceptance Model by Davis [61] as the main framework, with the actualized questionnaire developed by Perlman [175]. The experts were asked to answer the following questions about each perceived ease of use and perceived usefulness of the interactive tool for this purpose:

- As an instructor, how well does the app allow you to explore the details of the students' responses? Explain.
- As an instructor, how well does the app allow you to identify broad trends among the students' responses? Explain.

- As an instructor, how well would the app help you make instructional decisions related to student progress? Explain.
- As an instructor, how easy is it to use the app? Explain.
- As an instructor, to what extent does the app make it easier to assess student progress? Explain.
- As an instructor, to what extent does the app give you insight into the development of student ideas over time? Explain.
- As an instructor, to what extent does the app allow you to see the diversity in your students' responses and experiences? Explain.

Regarding perceived ease of use, all three experts felt the tool was intuitive, clear, and easy to use. Additionally, all felt the tool made it easy to see and explore student word use and usage changes over time, particularly for words of interest. The computer science professor stated, "Without looking at the detail, I can glance over the Word-Stream to catch up student's attitudes or behaviors over the course of the week. Positive words are expected to appear as students begin to learn new things and [are] excited...." This professor also noted that the right panel simultaneously "allows them to navigate through every single sentence."

On the other hand, the research methods professor felt that while the tool "makes it easy to identify trends in individual word usage," word frequency alone does not sufficiently support discernment of other trends in the student experience. Similarly, the STEM professor noted that the tool's focus on word frequency makes it useful for "assess[ing] students based on core words and relevant words," but "frequency of words may not be the only measurement to reflect the importance of a topic." In evaluating the perceived usefulness of the tool, limitations emerged, each relating to the issue of data granularity in determining the meaning. The experts noted from an instructional standpoint that it is difficult to evaluate the substance and emotion of student responses with an emphasis on single word frequency. Style of writing, for example, may reflect something about the student experience that is not reflected in specific word choices. To discern meaningful trends in the affective qualities of student responses or to develop an understanding of student challenges, successes, and misconceptions, patterns need to be pieced together from combinations of words. One expert suggested integrating a clustering or scaling algorithm to visualize the structure of responses at a different level of detail. Seeing relationships among combinations of words and how they change over

time would make it possible to discern meaning and emotion, which cannot be done at the level of single words, and without being as “time-consuming” as reading all of the students’ detailed responses. As one user pointed out, the tool shows trends over time based on the highlighted words, but hard to evaluate if this student made progress in the topic areas.

Despite these limitations, the research methods professor noted that the tool could be “really helpful... if you [have] a set of consistent questions you asked each week...” such as asking students about “concepts that [require] greater explanation or clarity each week.” This expert also notes that reflection questions would have to be crafted carefully in order to be used in the assessment. Both points reflect the inherent difficulty of evaluating student progress over time: identifying and describing trends in large amounts of qualitative data, such as written student assignments, to inform teaching has been a perennial issue for both instructors and qualitative education researchers.

4.5 Conclusion

In this chapter, we present our study for leveraging the potential of WordStream in support of exploratory analysis with temporal topic evolution and interactivity features through two case studies. In the first case study, we explored how to visualize qualitative education data using a small set of student responses collected over the course of a semester. Our resulting visualization interface and informal user study suggest that we have developed a tool with perceived ease of use and usefulness to show the student journal entries in a way that honors their qualitative properties and context-embedded nature, but some fundamental challenges remain. The evaluation from the user study illuminated additional concerns to address in future development. Specifically, both the WordStream and word cloud display processed words by frequency count, which is useful for summarizing the topics discussed by students but reveals little about how their cognitive ideas were developing individually or as a class. Future work might reveal cognitive or affective changes with the integration of criteria-based rubrics, clustering algorithms, co-occurrence analysis, or other methods that examine language in combination. While one solution to control for the variance from prompts would require the same prompt to be used repeatedly across the weeks to compare densities as a reliable quantitative measure of change over time. However, such shifts to standardize the collection of journal prompts would run counter to the broader objectives intended to leverage visualization to support the display and exploration of context-laden qualitative data used in classrooms. The need to structure data in order to work with current visualization

technology mirrors the broader challenges faced by qualitative researchers, such as the narrow availability of visualization methods that connect with qualitative data [212]. In the second case study, we developed an interactive exploratory visualization to analyze earthquake-related events and characterize the condition from social media posts. The general view and brushing/linking features support users in exploring and discovering patterns and relating events. The application of this technique to *VAST Challenge 2019: Mini-Challenge 3* dataset also confirms the design decisions of multiple linked views help users to convey the messages from the perspectives of community, location, and volume of related posts.

Whether WordStream is adopted as the driving visualization or supporting panel for the exploratory process, its incorporation shows a clear benefit of discerning the diversity and evolution of ideas over time. However, its applications might be limited by focusing on word occurrences rather than the underlying relationships among human ideas, which requires semantic topic modeling methods. One interesting insight we gather from the second case study is the complimenting information when data is represented in both timeline-based and network views, eliciting ideas for further data transformation approaches.

CHAPTER 5

QUANTITATIVE DATA DASHBOARD WITH MALVIEW: INTERACTIVE VISUALIZATION FOR COMPREHENDING MALWARE BEHAVIOR

The contents of this chapter have been taken from a previously published article:

NGUYEN, H. N., ABRI, F., PHAM, V., CHATTERJEE, M., NAMIN, A. S., AND DANG, T. MalView: Interactive Visual Analytics for Comprehending Malware Behavior. *IEEE Access* (2022). DOI: 10.1109/ACCESS.2022.3207782.

In this chapter, we explore time-series visualization under the lens of quantitative data with malware analysis. Malicious applications are usually comprehended through two major techniques, namely static and dynamic analyses. Through static analysis, a given malicious program is parsed, and some representative artifacts are produced without any execution, whereas the given malicious application needs to be executed when conducting dynamic analysis. These two mainstream techniques for analyzing the given software are effective in detecting certain classes of malware. More specifically, through static analysis, the patterns and signatures of the malware are exposed, helping in detecting any known malicious payload hidden in the code. On the other hand, behavioral and run-time execution patterns of software are explored through dynamic analysis. To ease the analysis process, a third analysis approach, known as the visual representation of the artifacts created by both static and dynamic analysis tools, would also be a supplementary asset for malware experts. This dissertation introduces MalView, an interactive visualization platform for malware analysis by which pattern matching techniques on both signature-based and behavioral analysis artifacts can be utilized to 1) classify malware, 2) identify the intention and location of the malicious payload in the artifacts, 3) analyze unknown malware (i.e., zero-day malware) by recognizing any unusual signature or behavior, and 4) explore the time dependencies and thus the system components affected or tampered by the underlying malware. The results of several case studies conducted in this work show that MalView offers more features and information compared to some other visualization tools, facilitating the malware analysis process.

MalView was published in IEEE Access, in 2022 (co-authored with Faranak Abri, Vung Pham, Moitrayee Chatterjee, Akbar Siami-Namin and Tommy Dang). The application and demonstration video of MalView can be accessed at malview.netlify.app and

malview.netlify.app/video.

5.1 Introduction

Malicious software applications, or malware, are the primary source of many security problems. These intentionally manipulative malicious applications intend to perform unauthorized activities on behalf of their originators on the host machines for various reasons such as stealing advanced technologies and intellectual properties, governmental acts of revenge, and tampering sensitive information, to name a few. Malware applications are complex software programs that are often obfuscated to disguise their main intentions and thus deceive network administrators and the underlying intrusion detection systems. Although such obfuscations can be captured, reported, and maintained in a repository as a reference for building better detection mechanisms, newer malware programs are constantly developed by professional hackers raising the challenging problem of zero-day malware detection [3]. As a result, in order to build an effective malware detection and defense system, it is crucial to understand each malware and comprehend its behavior through rigorous analysis.

There are two conventional approaches that are widely adopted for analyzing software programs: 1) *static* analysis by which the underlying software is parsed, and intermediate transformations of the underlying software are generated without actually executing the software program. For instance, a control-flow graph can be created to represent the execution control of the program under test; and 2) *dynamic* analysis by which the program under test is executed in a controlled environment (e.g., a sandbox) and the behavior of the program is observed under various environmental conditions for further analysis. For instance, a sandbox (e.g., Cuckoo [85]) can capture the processes that are created along with the files that are tampered with or modified during the execution of the program under test. These two conventional program analysis techniques (i.e., static and dynamic) are often complementary to each other, each targeting different types of faults or malicious activities in the program that is being analyzed. There is also a third “*hybrid*” approach that enables conducting both static and dynamic analysis of the program under test.

Although these conventional program analysis techniques are shown to be effective in comprehending static and dynamic features of the software under test, it is often time-consuming, labor-intensive, and technically challenging to build a customized analysis platform. Therefore, to ease performing such a complex analysis, some other analysis techniques with a smoother learning curve and faster comprehension of functionalities

of the underlying software under test should be developed for analysis purposes. The visual analytics approach is one of those possible solutions to facilitate the analysis process and efficiently and effectively showcase the processes involved in malware analysis.

This chapter introduces an interactive visualization platform, called MalView, for performing analytical reasoning of malware behaviors. MalView is an analysis-oriented development to our previously created malware visualization tool [164]. MalView emphasizes comprehensive understanding from visual analytics with in-depth, multi-faceted explorations of malware behavior and scalability to multiple malware families. The result of malware triage and analysis is significantly enhanced if a provenance of software artifacts can be identified, especially when specific attributes of suspected malware are used to identify similarities to a set of known malware artifacts, as shown by Casey et al. [31]. In light of improving malware analysis utilizing malware artifacts, the current prototype provides a detailed graphical representation for malware analysis to identify: (1) indicators of compromise and malicious activities, (2) tampering, modification, and possible damages occurred on the system, (3) the mechanic of how malware functions and infect, (4) the primary target of the malware, (5) the suspicious events occurred on the network, (6) the impact on the host and its registry, and more notably (7) the time and process dependencies occurred while executing the malware, the key feature of MalView.

Malware visualization systems can be categorized into three categories: Malware forensics, Malware Comparison, and Malware Summarization [215]. The work in MalView is under Malware Forensics and Malware Comparison categories: assisting the understanding of the behavior of an individual malware sample for forensics. By exploring the characteristics and relationships between the process and its dependencies and mapping them to visual features, MalView provides an interactive and intuitive platform to comprehend malware behavior towards the ultimate goal of generating rules and signatures for fully-automated malware detection systems. To demonstrate the effectiveness of MalView in identifying and interpreting malicious and suspicious activities of malware, the chapter reports the analysis of different families of malware namely: Remote Access Trojans (RATs), Backdoor, Ransomware, Behavioral, Email Flooder, and Hacktool. The results show that using MalView it is possible to quickly understand the main functionalities of the underlying malware without delving into a complex analysis of the static and dynamic analysis reports.

While conducting the case studies and inspecting some malware families, the authors noticed the different behavior exhibited by the same malware on different operating system (OS) platforms. As a result, each malware was executed and inspected on three

different Windows platforms: Windows XP, Windows 7, and Windows 10. Even though the execution of each malware was performed in a controlled environment, it was noticed that the newer platforms of Windows operating systems (e.g., Windows 10) were creating more system and kernel-level processes making it harder to thoroughly inspect and analyze the exact flow of each malware on these recent versions of platforms. As a remedy for such problem, it is suggested to apply additional filtering mechanisms in order to analyze each malware and its processes thoroughly. This chapter makes the following key contributions:

1. It introduces MalView, a malware visualization tool to enable analytical reasoning of malware behaviors.
2. The MalView visualization tool visualizes the output of several dynamic and static analysis tools.
3. The tool also integrates the output of many anti-virus tools using their Application Programming Interface (API) to provide additional insights for each malware.
4. The chapter demonstrates the efficiency and effectiveness of MalView through several case studies conducted on a set of the family of malware.
5. The chapter also compares the behavior of each malware when executed on three different Windows platforms (i.e., XP, 7, and 10) in order to recognize the impact of environmental settings on malware comprehension and analysis.

Organization of the chapter. The rest of this chapter is laid out as follows: The state-of-the-art of malware visualisation is presented in Section 5.2. Section 5.3 gives an overview of the data collected and feed into the visualization and introduces the system and visualization tasks that guide the system design. Next, section 5.4 elaborates on the visual components with interactivity and highlights the key features of MalView. This section also presents the analysis performed using MalView on a set of family of malware types. The influence of the running platforms on malware behavior is articulated in Section 5.5. A feature-based comparison of MalView and some other malware visualization tools is demonstrated. Section 5.6 concludes the chapter and highlights the future research work.

5.2 Background and Related Work

The malware analysis methods can be broadly categorized into static vs. dynamic analysis [89]. Many of these approaches utilize visual representation to enable the analysts

to visually capture general activities related to malware from a large number of data files or logs which are infeasible to digest in text or binary format [82].

5.2.1 Signature-based Features/Static Analysis

Panas [171] visualized software binaries in order to demonstrate malware samples. In their approach, they first disassemble the file to obtain the Abstract Syntax Tree (AST) and then provided the intermediate representation of the file by using ROSE [120], an open-source compiler. Visualizing the signature of a set of different malware families, they were able to show the changes in different versions of a malware family. Also utilizing visualizations in dynamic malware analysis, Grégio et al. [82] proposed a solution with two interactive visualization tools. The two visualization prototypes are a timeline with a magnifier and a spiral view of the malicious activities. The first tool provides analysts with views of the malware activities over time. While the time selection for the x-axis is similar to ours (and many others), the uses of colors and what is to be presented in the y-axis are different. They used the y-axis to represent activities and colors to different processes or services involved by the malware execution. Each event (an activity at a timestamp of an involved process) is presented by a circle connected by a line, which represents changes over time. This presentation leads to the visual cluttering issue, especially when malware does many different activities in a short time interval [232].

Gove et al. [80] presented their tool Similarity Evidence Explorer for Malware (SEEM), which compares a focal sample of malware with other malicious samples in the database. The malware features are grouped into nine categories, and feature similarities are visually presented in three ways: 1) histogram, 2) Venn diagram list, and 3) a feature matrix. The histogram utilizes the Jaccard similarity of the features of the focal sample with the other samples. In contrast, the Venn diagram is more granular and shows information of overlap, strict subset, and disjoint features. The feature matrix highlights the specific features present in the analyzed sample. Long et al. [127] proposed a versatile and instinctive technique to identify a given malware file from its image sets. The authors argued that the desktop icons are one effective social engineering attempt employed by some malware developers. The victim clicks on the icon resulting in the execution of the malware. Hence, comparing a new malware based on its image with a previously known malware database results in effective malware identification. Extracted greyscale malware images are shown using a *force-directed graph* [210, 162], which is essentially a similarity network of sample malware images computed with the nearest neighbor index. The visualization tool shows hash values of the executable,

and upon clicking on the values, it draws the similarity network graph, and it provides zooming functionalities for multiple hops of each node.

5.2.2 Image-based and Dynamic Analysis

While static analysis is computationally efficient, its performance could be impaired by packed or encrypted malware. On the other hand, dynamic analysis analyzes actual behaviors from the malware while it is running, so it is more efficient [107]. In their work [195], Shaid and Maarof proposed a method to generate images representing malware API calls. First, the API calls are monitored in the malware behavior capturing step. These calls are then sorted from malicious to less malicious. Finally, each API call is assigned a color depending on its maliciousness level. Similarly, Kancherla et al. [107] proposed to convert the malware into a gray-scale image called byteplot. They then used machine learning (ML) methods (e.g., Support Vector Machines) to analyze the low-level features (e.g., intensity and textures) extracted from the resulted images. Regarding ML approaches, LeDoux and Lakhoria [122] presented that ML has a natural fit with malware analysis, where ML operates by rapidly learning, discovering inherent patterns and similarities in the corpus.

With image-based malware classification, O'Shaughnessy [169] utilized the space-filling curves approach to formalize a scalable solution for classification ambiguity among anti-virus programs. Donahue [65] proposed another idea of using Markov Byte Plot [106] to convert Portable Executable (PE) files into truecolor (defined by red, green, and blue (RGB) color components) images that help to highlight the differences between the packed and unpacked malware. Another common approach is to convert malware PE or binary files into images and analyze the resulted images. There are various ways to turn the malware into images and different methods to analyze the produced images. For instance, Han et al., [90] proposed a three-step approach to analyze malware in this direction. First, the opcode sequences of the malware are extracted in Step 1. Step 2 generates an image with both width and height are of 2^n , where n is a user-defined number. Next, this step applies a hash function, such as SimHash [33], to each of the extracted opcode sequences to generate a pixel with corresponding x-y position and RGB color. Finally, similarities between the resulted images are calculated in Step 3. In their extended version of this approach [89], they also incorporated dynamic analysis to filter for essential opcode sequences.

Miles et al. [144] presented VirusBattle, a system equipped with intelligence navigation and visualization to mine and to discover interrelationships between malware instances automatically. This system provides two primary analyses: 1) a program's

dynamic trace tree and 2) a scalable method of discovering shared Computed Semantics artifacts among instances of malware. VirusBattle analyzes the interrelationships over many types of malware artifacts, including the binary, code, code semantics, dynamic behaviors, malware metadata. Shaid and Marrof [194] proposed the method of presenting the behavioral pattern of malicious files using a Hot-to-Cold color ramp. As the malware runs, the user-mode API calls are captured, then ordered and grouped based on their maliciousness. This behavior-to-color map of the malware helps visualize when and in which order a malware sample performs malicious activities during execution.

Besides software systems, research in hardware advancement has introduced many approaches that facilitate malware analysis to build a transparent dynamic analysis system. In terms of hardware virtualization extensions, Dinaburg et al. [64] proposed Ether, an application that remained transparent and defeated a large percentage of obfuscation tools. Later, Lengyel et al. [123] built DRAKVUF on a similar virtualization extensions approach and provided greater insight into the execution of the system to trace system execution for malware analysis.

This visual analytics approach in MalView can benefit the branch prediction in dynamic environment analysis in GoldenEye by Xu et al. [226] and the analysis of sequences of API calls in VECG by Alaeiyan et al. [6]. The interactive visual representations can expedite the process of proactively detecting environment-sensitive and context-based behaviors, where “human-in-the-loop” can accelerate early stopping and quickly capture patterns that emerged from the API call sequences.

5.2.3 Hybrid (Static and Dynamic) Analysis

Most static approaches focus on comparing, clustering malware instances, or classifying if a new sample belongs to a known family of malware. For example, Paturi et al., [172] used Pythagoras tree to represent the similarities in codes between malware. The similarity metrics might be “Cosine similarity” or “Normalized Compression Distance.” The hierarchical structure of the Pythagoras tree is characterized as the distance between nodes at each lower depth of the tree is reduced by $\sqrt{2}/2$. Thus, the tree helps bring malware with higher similarities into clusters as leaf nodes with shorter distances stay close to one another.

Anderson et al. [10] presented a malware classification system that works based on the combination of static and dynamic features. For static feature extraction, they used three sources, including 1) the binary file, 2) the disassembled binary, and 3) the control flow graph of the disassembled binary file. For dynamic feature extraction, they used dynamic instruction sequence and the dynamic call sequence. They tested their

system using a large malware dataset and achieved 98.07% accuracy with the combined static and dynamic features. They also achieved a 96.14% accuracy by using only static features. Yoo [229] designed the visualization based on the belief that malicious content in an executable file has a unique feature called SOM (Self-Organizing Map). By calculating the SOM and visualizing a specific executable file, the potential portion of the malicious content can be determined, and by checking the generated pattern, the malware family can be detected.

Saxe et al. [189] developed an interactive visualization system for comparing malware samples in a dataset using the extracted features. Based on the presence of the system call sequence, the similarity matrix for the malware dataset is generated. This system also provides a comparison view among malware samples based on their malicious activity. On mobile computing platforms such as Android devices, Jenkins and Cai [102, 103] explored Inter-Component Communications (ICC) via interactive visual explorations, showing thorough ICC comprehension and security vulnerability inspection, revealing the malicious behaviors that were normally hidden to users.

5.2.4 Visualization Tools and Analysis

Analyzing malware through visual behaviors has been studied with the aim to observe the overall flow of a program, discover malicious patterns, and quickly assess the nature of the malware sample [214, 215]. Wagner et al. [216] proposed KAMAS, a knowledge-assisted visualization system for behavior-based malware analysis, which visualizes API call sequences gathered during the execution of malicious software. Our approach aligns with this direction, but we shift the focus on the analysis side with different malware families and the influence of operating systems on malware behavior. In particular, the design decisions and techniques in MalView are applied in the malware analysis domain and derived from visualization principles for time-series data, which is the collection of observations through repeated measurements over time, including but not limited to numerical, geolocation, and text data [49, 53, 161]. Using Ether [64] as the monitoring platform, Quist and Liebrock [182] propose a directed graph structure of all the basic blocks of an executable with a navigable interface to explore the code structure. Treemaps and thread graphs are also visualization techniques that show usefulness in detecting maliciousness of software and in classifying malicious behavior [207]. The classification decisions can be supported by the visual analytics solution provided by Angelini et al. [13] to provide the user a better understanding of such decisions and the possibility of changing the classification results. Visual analytics approach, when combined with predictive analysis, can project potential threats or detect malicious attacks

for securing efficient manufacturing automation [121].

Conti et al. [40] designed a system for file analysis with these features: 1) Analyze undocumented file format, 2) Audit files for vulnerabilities, 3) Compare files, 4) Crack, Cryptanalysis, and Forensic analysis, 5) Identify unknown file format, 6) Malware analysis and 7) Reporting. Their system is an extension to the hex editor and consists of both textual and graphical visualization. Quist and Liebrock [182] developed a tool called VERA (Visualization of Executables for Reversing and Analysis) that can be used for visualizing the structure and flow of an executable file, including memory reads and writes. Later, they extended their work [183] by adding more reverse engineering tools and providing more testing case studies in detail.

Trinius et al. [207] used two different approaches for malware visualization. They first generated an XML file containing dynamic analysis information of the malware sample using *CWSandbox* [47], including 1) loaded system libraries, 2) outgoing and incoming network connections, and 3) accessed or manipulated registry keys. Using the XML output, they visualized the key feature using two techniques: “*treemaps*” and “*thread graphs*”. They argue that these two methods are complementary and, using these two visual representations, can effectively help detect the malicious behavior of the given malware and identify the malware family. They tested their proposed approach by executable and non-executable (PDF format) malware samples.

Gregio and Santos [83] developed an interactive timeline tool for visualizing dynamic malware behavior using various techniques [232]. They ran the given malware in a controlled environment and captured its behavior using a modified version of BehEMOT [81] (a malware behavior monitoring tool). They captured high-level activities such as file write and delete, process creation and termination, registry reads and writes, mutexes and network operations, and system calls using System Service Dispatch Table (SSDT) hooking, which operates at the kernel level. In addition, they used identification labels provided by VirusTotal [94].

An interactive visualization tool called MalwareVis is introduced by Zhou et al. [232] for malware dynamic analysis with a concentration on network traces including the total number of packets, size of the transmission, number of streams, and the packet trace’s duration. They ran the malware in a controlled environment and captured these network traces by using packet sniffer software, where the output packet capture (PCAP) files were used for visualization. They used table views and shape views for representing the features that allow the user to browse, filter, and compare different types of malware. Cappers et al. [30] also utilize PCAP to discover the patterns in traffic to explore the intrusive behavior from malware activities.

5.3 Methodology

5.3.1 Capturing Dynamic Behavior Using ProcMon

Dynamic analysis aims at studying the behavior and actions of malware sample when it is executed. This technique analyzes malware and returns the collected information of such behavior and actions for further processing or analysis [215]. Assuming that the malicious sample does not employ any anti-forensics guards, in this chapter, the Windows Sysinternals Process Monitor [138], or Procmon, is employed to capture the run time behavior of malware during execution. MalView visualizes the outputs and traces produced by Procmon rather than explicitly executing a given malware directly. During the dynamic analysis of malware execution, Procmon can capture five types of events that the Windows-based malware interacts with the host system: 1) file system, 2) registry, 3) network, 4) process and 5) profiling.

While capturing dynamic behavior of malware, it is important to use a proper Procmon filtering to avoid capturing unnecessary information from the normal execution of the system. Furthermore, even when the underlying system is idle, it has numerous background processes running that can be captured by the Procmon. As a result, the authors filtered out the activities by capturing suspicious processes only represented by functions commonly encountered by malware analysts [99, 100]. Furthermore, they excluded the default system operations such as Procmon, Autoruns, Sysmon from further visualization and analysis.

5.3.1.1 Data Attributes

Procmon provides records of Windows activities through the low-level system events, where thousands of events are generated every minute. The standard output in Comma Separated Value (CSV) format from Procmon is used as the primary input for visualization components in MalView. One row in the CSV log file demonstrates one specific event and comprises of these major attributes [185]:

- Time of Day: The timestamp of the day when the event occurred.
- Process Name: The name of the process – active executable, performing the operation.
- PID: Process identifier (ID).
- Operation: The name of the executing operation.

- Path: The path to the target object being operated on. This field can be empty, depending on the operation/process.
- Result: The result of the operation. The values for this field include success, denied, or access.
- Detail: The additional notes about the event.

5.3.1.2 Event Categories

The log file output from Procmon contains five major types of process activities, which are color-coded in our framework.

- Registry: Events of registry operations, such as querying and enumerating keys and values.
- File System: Events related to operations on local and remote storage and file systems.
- Network: Network activities, including TCP and UDP.
- Process: Events of process/thread, such as process creation, start, and exit.
- Profiling: Events for every process in the system in terms of memory used, kernel and user time charged, output as a log for the profile.

5.3.2 MalView: System Overview

MalView is aimed at accelerating malware analysis and integrating visual analytics to enable interactive data exploration and malware behavior comprehension. Figure 5.1 depicts the architecture of MalView. The flow of information in MalView is as follows: 1) It uses a data provider in dynamic analysis, where the malware sample is executed on a host system, then the data provider logs relevant information into execution traces. 2) MalView takes in the raw data captured by the data provider, extracts the information, and maps them to visual features. 3) MalView explores the relationship between each process and its dependencies. To the best of our knowledge, this feature has not been taken into account in previous work, not only the malware as an individual but also its interactions with the system and the artifacts created.

The MalView prototype provides visual representations for system and malware activities captured by Procmon [138] utility. In the context of malware analysis, four important system-level activities are of utmost importance that need to be captured, namely

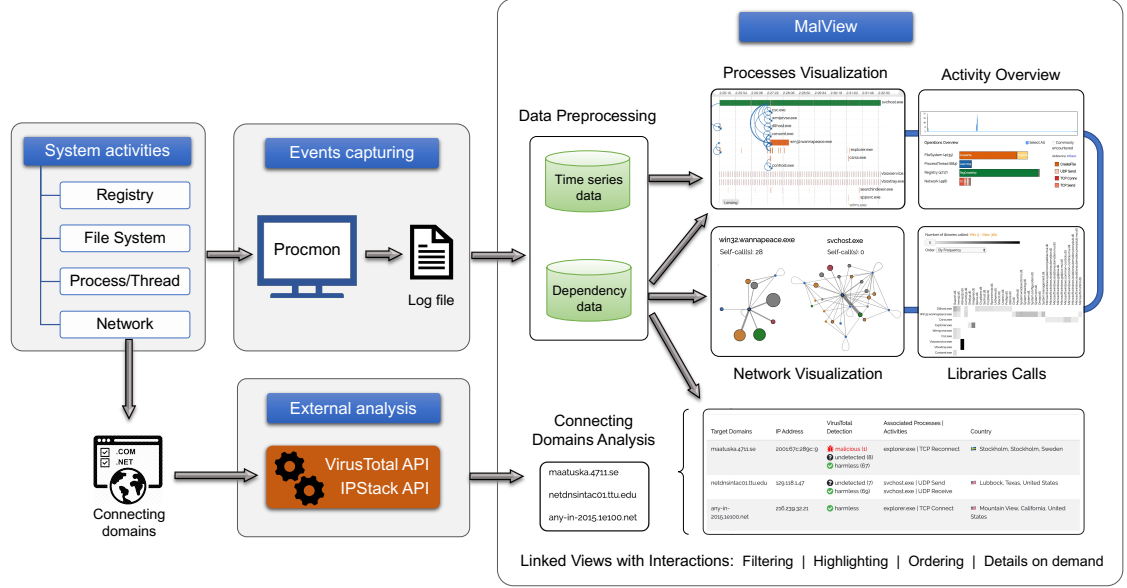


Figure 5.1: The schematic overview of MalView framework for analyzing the dynamic behavior of malware. The visualization provides linked views and supports interactive features, such as filtering, highlighting, ranking, and details-on-demand.

registry, file system, processes and threads, and network activities. These are four major categories that are highlighted by InfoSec [99, 100] as an indication of malicious activities. The processes and events related to these four activities are captured by Procmon, filtered, and then fed to MalView.

MalView provides an analysis of linked views with interactions for users to gain comprehensive insights into malware behaviors within the system. Details of MalView visual components with their corresponding interactive features are described in the following section, MalView: Visual components.

The tool MalView is developed as a web-based application using JavaScript and D3.js library created by M. Bostock et al. [26]. The primary goal of MalView is to provide an interactive visualization platform that demonstrates the malware behaviors and interactions within the system. The captured events are presented in multiple perspectives: in a temporal manner of processes and function calls between them, the dependency graph between a process and the objects it operates on, including registry, system files, network addresses, and dynamic-link libraries. The platform gives classification of malicious or benign connecting domains with further analysis. To meet the primary goal, MalView implements the analysis tasks below, based on the analysis task types for employing information visualization systems [9]:

- **T1 Provide a comprehensive overview of system activities.** The visual design should present the general distribution of activities chronologically to facilitate the initial summary based on the selected malware.
- **T2 Display details-on-demand for activities and interactions.** The user can get a close-up look at an entity or select an activity to view its event data. The system should show the information in a deeper level of supporting details that accompany the interactions among different processes.
- **T3 Characterize data distribution for processes and their dependencies.** In addition to displaying the detailed information of processes on demands, it is also important to show the distribution of temporal patterns of processes and simultaneously use that as the context to explore their dependencies. To this end, the system should show the groupings of operated objects (e.g., dynamic-link libraries) based on their similarities in features.
- **T4 Present the associations: relationships among processes and function calls between a process and its dependencies.** To characterize the complex associations between entities within the system, the system should show the relationships caused by interactions among processes and function calls from a process to its dependencies.
- **T5 Highlight critical activities in context.** Here, *critical* is defined in context: For the timeline as a whole, MalView should allow user to zoom into the time interval that captured the most active interactions of the malware. For malware activities in particular, the system should incorporate filter-based feature to highlight the commonly encountered malicious types, besides the original representation.
- **T6 Order the entities based on dependencies characteristics.** A specific ranking order along a data dimension be of tremendous help in the arrangement of visual components to convey important characteristics and allow the user to focus on the top essential entities.
- **T7 Classify malicious vs. benign activities.** Another key to understand malware forensics is the ability to show the malicious and benign activities. The system should be able to classify the level of malice that corresponds with the malware sample captured.

5.4 Results

5.4.1 MalView: Visual Components

Taking into account the mapping to time, the associations between processes and dependencies, and guided by the designed tasks, we designed the user interfaces of MalView. Figure 5.2 depicts the main modules of MalView for supporting users in comprehension and visual reasoning of malware activities and interactions.

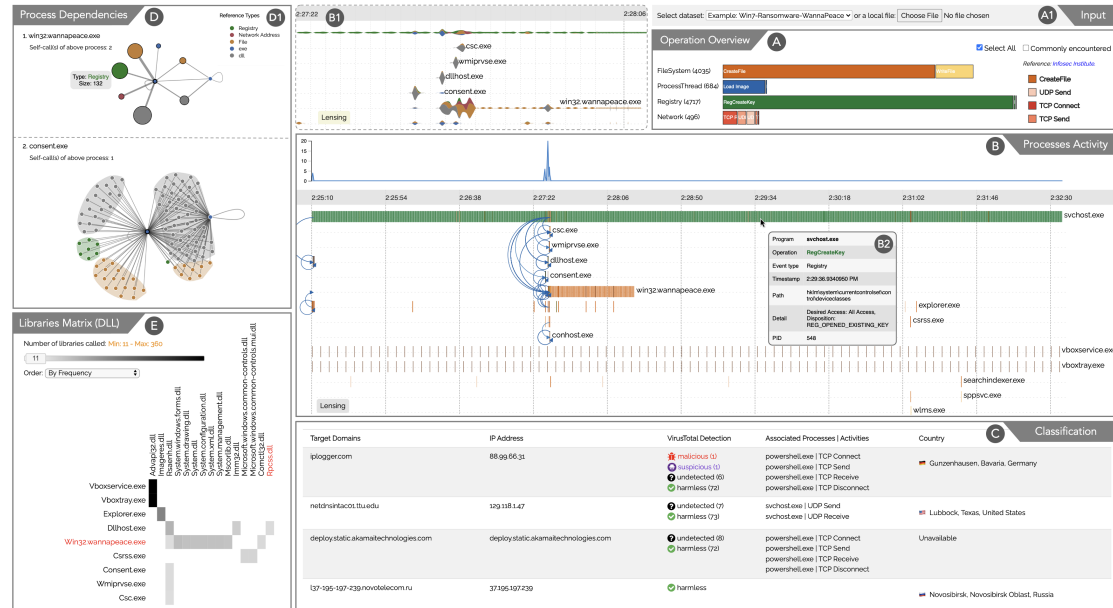


Figure 5.2: The MalView system contains the following main views: *Input and Operation Overview* (A), *Processes Activity* (B), *Classification* (C), *Process Dependencies* (D) and *Libraries Matrix* (E). The *Process Activity* view has another mode of showing the *referenced operated object* (B1), with feature of lensing for zooming in upon a particular interval in both viewing modes. Detail-on-demand is provided upon mouse interaction on selected item, as in the example in panel (B2).

5.4.1.1 Input and Operation Overview

MalView provides the options for users to select input data from a default set of data samples or from their local machine, as depicted in Figure 5.2, panel (A1). The default dataset contains more than 60 samples of primary input data, helping users to familiarize themselves with the system and explore how the tool works. In addition to the default dataset, MalView allows users to use and analyze the output log file from running Procmon on their machine via the “Choose file” button for uploading the file for direct analysis.

After the input file is uploaded, the *operation overview* (A) shows how the operations are categorized and allows user to observe the the prevalence of event types (visualization task **T3**). Each event type is represented as a rectangle, stacked horizontally by its category in a bar chart visualization. There are four color hue representing four categories: yellow for File System, blue for Process and Thread, green for Registry, red for Network. We employ the color coding based on the category the event type belongs to and incorporated it with the statistics of the amount of total corresponding function calls during the monitored period. An individual event type is mounted with interactivity: it acts as a button providing filtering upon mouseclick, the result of which is shown directly on the below adjacent panel, *processes activity* (B)

A special group of existing critical operations, defined by “Commonly encounter” from InfoSec [99, 100], is shown on the right of panel A. All the available operations captured that match the commonly encounter criteria are presented. This list serves as a selection box for highlighting critical activities in both *operation overview* and *processes activity* (visualization task **T5**).

5.4.1.2 Temporal Patterns

Building upon the visual information mantra by Shneiderman [197]: “Overview first, zoom and filter, then details-on-demand,” the *process activity* in Figure 5.2(B) is designed to explore the temporal patterns from the system’s low-level events along with inter-process communications. The timeline is presented horizontally from left to right, while the processes are listed vertically. Besides the operations executed by a process itself, there are interactions between two processes, such as one creating the other with its primary thread, demonstrated by the arc connecting the two. On top of panel B is an area chart showing the arc distribution, providing the overview of the function call frequencies (visualization task **T1**).

Each process is associated with an aligned set of events executed by the process itself. An individual event is represented by a thin vertical bar, color coded by its event type, which is introduced in section 5.3.1.2 and presented in panel A. These small, thin bars are presented with 50% transparency so that if multiple events appear at nearly the same time, the color will add up on display (visualization task **T3**); therefore, users can see that the calls are busy there and there is a chance for anomalies detection at these spots (visualization task **T7**).

The interaction arc starts from the parent process (the one that initializes the call, or the source) and ends at the child process (the destination, or the target of the call) (visualization task **T4**). The interactions here are the typical events of processes and

thread, as specified in section 5.3.1.2; hence they have the blue color of process and thread category. One of the most common events in this category is Process Create, in which a process creates a new process and its primary thread. Besides the source-target interactions, MalView also supports to visualize the call-to-self events (or the loops). In this case, the process is both the one that initializes and the target of the call.

MalView supports details-on-demand in terms of process detail, event call detail, filtering calls related to one specific process, and zooming in a period (visualization task **T2**). The details of an event can be shown on the tooltip by mousing over the corresponding bar, including process name, operation, event type, timestamp, process ID, and additional operation-specific information about the event, as shown in Figure 5.2(B2). Similarly, the detail of process interaction is displayed on the tooltip by mousing over the arc, providing information on the source process, target process, and the event type of the call. For a particular process, users can choose to observe only the call originated from or to this process by a simple mouseclick on that process. The zooming feature for the arc distribution (visualization task **T5**) will be presented with a case study in section 5.5.1.3.

Operated Objects This *processes activity* panel supports the detail view by a magnification feature called “Lensing” (visualization task **T2**). When this feature is enabled, hovering along the timeline will expand the current window at that time step. For example, panel (B1) in Figure 5.2 presents the “Lensing” feature for the interval of 2:27:22 to 2:28:06. Here, the view shows another mode of presenting the referenced streamgraph rather than individual events. We utilize the event categorization that revolves around five key types: registry, file system, network, process, and profiling, to determine the operated objects. Since profiling operation can be less informative about process activity and more about kernel time and memory used, we exclude profiling from the scope of our operated objects. In addition, dynamic-link libraries that contain code and data that can be used by more than one program at once are also indispensable from the analysis process. These considerations lead to our final operated object list: *registry*, *network address*, *system file*, *exe* (executable file), and *dll*, as shown in panel (D1). That serves as a reference to both panel (B1) and, later, *process dependencies* in panel (D).

5.4.1.3 Malware and Classification

Figure 5.2(C) presents the classification for malicious or benign activities of the captured log file produced by Procmon (visualization task **T7**). Aligning with the primary aim of providing a visual analytics tool and platform to demonstrate malware’s static and

dynamic behavior, MalView captures the results provided by the integrated APIs and visualizes them to the end-user. MalView incorporates a number of APIs such as VirusTotal API and inherently relies on the output produced by these APIs. We investigate the target domains that the network activities are connected to. The extracted information for each connected domain contains its Internet Protocol (IP) address, the detection classification results, the associated process and activities related to the domain, and lastly, the country to which the server is hosted.

The API automatically scans a given malware, and their patterns are automatically compared with more than 70 servers and databases. The classification result consists of four categories: malicious, suspicious, undetected, or harmless, each indicated by the number of detections found corresponding to the targeted domain. Spring et al. [199] discussed that the malicious domains are attempts to connect with a command and control server or dropbox and are expected to behave differently from a typical phishing or a drive-by-download malicious site. In MalView, this list of connecting domains is ordered by the variety of the outcomes of each domain (visualization task **T6**). Figure 5.3 demonstrates the analysis summary of *TeeracB* malware on Windows 7. One malicious domain is detected, named “maatуска.4711.se”, connected by the “explorer.exe” process with “TCP Reconnect” activity.










Target Domains	IP Address	VirusTotal Detection	Associated Processes Activities	Country
maatуска.4711.se	2001:67c:289c::9	 malicious (1)  undetected (8)  harmless (67)	explorer.exe TCP Reconnect	 Stockholm, Stockholm, Sweden
netdnsintac01.ttu.edu	129.118.1.47	 undetected (7)  harmless (69)	svchost.exe UDP Send svchost.exe UDP Receive	 Lubbock, Texas, United States
any-in-2015.1e100.net	216.239.32.21	 harmless	explorer.exe TCP Connect	 Mountain View, California, United States

Figure 5.3: MalView analysis summary of TeeracB malware on Windows 7.

5.4.1.4 Process Dependencies

This *process dependencies* view (Figure 5.2(D)) presents an in-depth analysis of each process in the system, where one process can operate on many types of objects, as introduced in section 5.4.1.2 and shown in panel (D1). The visualization task **T4** is actualized as presenting the one-to-many relationships between the process and its dependencies. In addition, as the number of dependencies increases in cases with complex activity, we need a way to handle visual clutters by reducing the number of visual

elements while preserving the structure. For these reasons, we employ 1) the force-directed layout with node-link diagram to demonstrate the relationships and 2) the node bundling technique [133] incorporated into the force-directed layout to reduce visual clutter by node aggregation. Force-directed layout has been explored in many efforts, such as [162, 157, 210], to represent the even distribution of nodes and links and speed up spring force calculations.

The *process dependencies* panel contains multiple windows; each corresponds to one single process, ranked by the degree of that process node (visualization task **T6**). In each window, aside from the main process node (positioned in the center of the graph with thick, dark stroke), each node can be in one of the two states: individual or bundling. The individual state corresponds to each node representing one object operated on by the process. The bundling state leverage the node bundling/aggregation technique [133], as shown in the top panel of (D), where each node encompasses multiple objects with the same type and connection. The size of this bundled node is proportional to the number of the individual nodes it comprises (visualization task **T3**). Mouse-clicking on a bundled node transforms itself into a set of individual nodes bounded by a convex hull, as shown in the lower panel. These two states can be switched back and forth by a single mouse click on the bundled node or the convex hull surrounding the internal nodes. Besides the source-target type of connection, the graph also presents the available call-to-self events (the loops) of each process, in accordance with the *processes activity* panel in Figure 5.2(B).

5.4.1.5 Libraries Matrix

Figure 5.2(E) describes the dynamic-link library (DLL) calls by each process (visualization task **T4**), supporting users to detect the abnormal frequency patterns (visualization task **T7**). These are the Windows API calls to the libraries that are part of the Windows operating system, not to be confused with the one calling VirusTotal/IPStack API for scanning connected domains, as presented in Section 5.4.1.3. System activities may involve multiple library calls from one process or a common library providing resources for various processes. To represent vast number of relationships between processes and libraries, MalView utilizes an interactive heat-map matrix to prevent cluttering in contrast to conventional node-link graph visualization (visualization task **T3**). In the matrix, each cell value is color encoded by the gray color scale, in which darker presents frequent calls while lighter is rare calls. There are several criteria for ranking processes (rows)/libraries (columns): by similarity, frequency, or the number of different libraries called.

5.4.2 Case Studies

In an effort to provide its users with a safe and productive experience, Microsoft provides information about malware and unwanted applications affecting its operating systems online [136] and details about these in its documentation platform [137]. Microsoft [137] classifies malware into 13 categories: 1) Backdoor, 2) Downloader, 3) Dropper, 4) Exploit, 5) Hacktool, 6) Macro virus, 7) Obfuscator, 8) Password Stealer, 9) Ransomware, 10) Rogue security software, 11) Trojan, 12) Trojan clicker, and 13) Worm. Furthermore, Microsoft also provides a tool to search for current cyber threats, viruses, and malware in its online platform called Microsoft Security Intelligence (MSI) platform [135].

MalView can be utilized in different settings. 1) When the objective is to comprehend malware functionalities and not detection, 2) when a new malware application (zero-day malware) is developed and not detectable by any tool (due to lack of profiles and signatures), 3) when the objective is to classify a family of malware and then employ a set of generic solutions and remedies to address each class of malware, and 4) when new malware is developed, and we are interested in investigating whether it follows some existing known malicious patterns or not (i.e., labeling malware type). Accordingly, if there is an incident report about zero-day vulnerability where there is no clear patching solution developed, MalView can help us to analyze and comprehend the malware with zero-day vulnerability and thus enable us to identify patches or solutions better. To demonstrate the usability of MalView in analyzing malware software visually, we conducted a set of case studies in which the output and behavior of the selected malware were captured. Due to the space limit, we capture and present the processes involved in seven malware, namely 1) Backdoor, 2) RemoteAccess, 3) Behaviour, 4) Ransomware, 5) EmailFlooder, 6) Hacktool, and 7) Trojan (Info stealer). The following sections demonstrate the applications of MalView to several of these malware types.

5.4.2.1 Experimental Setup

The malware experimentation setup needs an isolated and controlled environment so that the malicious code does not propagate or infect other entities in the network. This clean and isolated environment also helps to identify the changes and possible tampers in the system due to the malicious activities of the malware specimen. For this work, we installed three different Windows systems on an Oracle Virtual Box: Windows XP, Windows 7, and Windows 10. The windows defender services, windows security services, firewalls, and other automatic security updates were disabled on each of the virtual OSs

to prevent any interruption during the malware sample’s execution and capture all the traces of their dynamic behavior. To capture the interaction between the malware and each host system, Procmon was installed on all environments. More specifically, all the user applications on the virtual OS were closed, the malware process name was added to the monitor filter to capture only the events of the malware executable. Then the executable was run for two minutes before saving the time-ordered system activities from Procmon and fed to MalView.

Since MalView depends on the output of Procmon, the amount of information it visualizes depends on how long Procmon is executed. The execution time also shortens the amount of data captured by Procmon. According to our experience with MalView, a larger and more complex output and traces produced by Procmon makes MalView less effective since the visualization needs to capture a vast number of processes and events. However, a key feature of MalView is to offer different levels of abstraction and complexity. If we adjust the window width (interface size) and rerun the sample, the visual components would readjust to fit the new window size. More specifically, the execution time depends on how large the malware sample is, ranging from 0.3s to several seconds.

5.4.2.2 Remote Access Trojan (RAT)

Remote Access Tools are useful applications to provide administrative assistance to the end-users remotely. However, these pieces of software are increasingly abused by adversaries to gain control over the target systems and are referred to as Remote Access Trojans (RATs). RATs are distributed through email attachments or as a patch with pirated software to infect the target in order to gain administrative control. Once the target machines are infected, RATs have complete control over the victim system to perform malicious activities, such as password sniffing, keylogging, track file transfer information, control the system by issuing shell commands, or even propagate some other malwares/viruses. RATs are particularly hard to detect, as they execute legitimate operating system processes resembling the behavior of other commercial remote access tools, and they usually do not show up as running tasks. Besides, there are tools that enable obfuscation on a given application and produce obfuscated malicious applications. Using various obfuscation methods, along with managing resource utilization, RATs can remain undetected. According to the October 2018 Global Threat Index [35] published by Check Point, RATs are ranked among the top 10 “most wanted” malware.

We captured the run time behavior of RATs on different Windows and visualized the behavior using visualization tool MalView. The live malware sample was downloaded

from public malware dataset *VirusShare* [42]. According to a multi-scan report from Virustotal [94], this sample has a community score of 66 out of 70, i.e., out of 70 detection engines, 66 could identify it as a malicious executable. Figure 5.4 shows the detail analysis performed on an RAT sample using MalView. The malicious indicators presented by MalView are as follows:

- **Process:** The malicious executable spawns processes like *explorer.exe*, *wscript.exe*, and *svchost.exe*. The execution of these processes indicates that the RAT program is trying to start a command prompt and then run some scripts to start a session to monitor the process remotely.
- **Registry:** The sample RAT performs a large number of registry operations, including the creation of registry keys as well as a query of the registry entries.
- **Files:** The malicious PE performs a large number of various file operations, including the creation of new files and mapping file systems.
- **Network activity:** The sample RAT does not demonstrate any significant number of TCP/UDP requests.

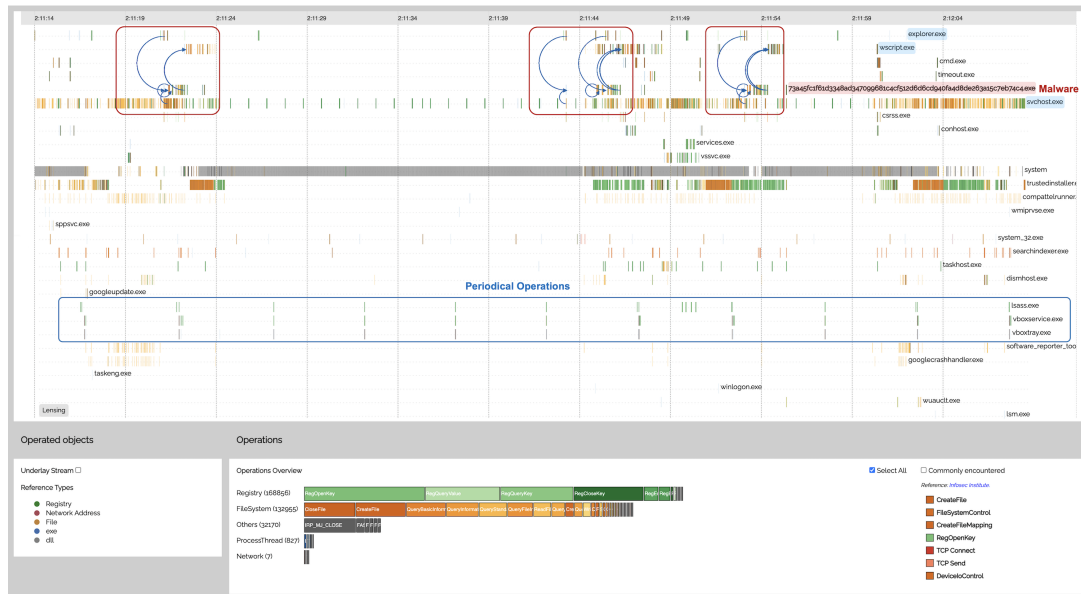


Figure 5.4: MalView analysis summary on RAT, with the filter set on displaying the interactions with RAT only: There are multiple and repeated function calls between the process corresponding to the RAT and *explorer.exe*, *wscript.exe* and *svchost.exe*. Patterns of periodical operations are also presented, such as of Local Security Authority Subsystem Service *lsass.exe* or Virtual Box’s *vboxservice.exe*.

Besides the malware-associated events, MalView is also able to capture the recurrent pattern of periodical operations, such as the system process of Local Security Authority Subsystem Service *lsass.exe* or Virtual Box’s *vboxservice.exe*. The influence of running platform will be discussed further in Section 5.5.1.

5.4.2.3 Trojans

A Trojan is a type of malware that pretends to be a benign program, but after installation, it executes hidden code and then performs malicious activities such as deleting or tampering with data, stealing information, running some other scripts, and creating backdoors. In general, it enables the attacker to access the victim’s system, and these types of malware are not able to replicate themselves [110].

Sample Trojan A sample of Trojan¹ was obtained from VirusShare [42]. The output file containing all the processes was created after running the malware in a controlled environment using Windows 7 as its platform. Figure 5.5 shows the MalView output for this malware. We applied lensing on the critical period to view the activity details. We chose four important processes based on the dependencies, including *cmd.exe*, *tmp.exe*, *reg.exe*, and *timeout.exe*.

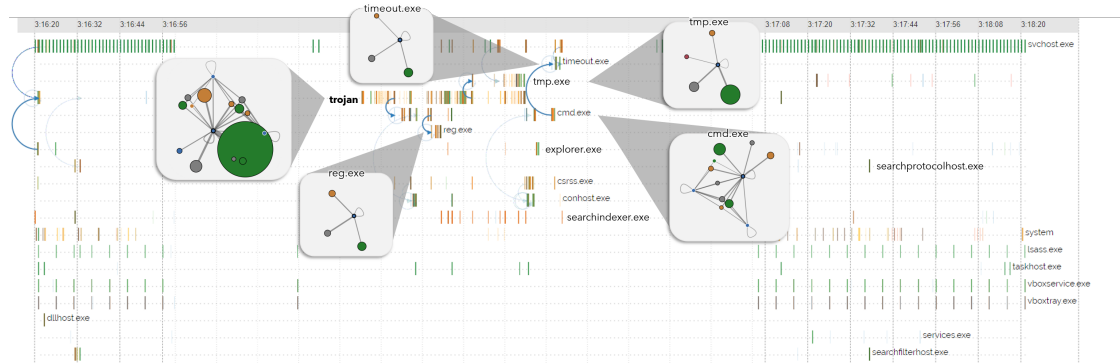


Figure 5.5: MalView visualization for a sample Trojan. User can request to overlay the networks of suspicious processes (which contain the self calls).

By clicking the name of this malware on MalView Process Activity window, we can observe that this executable file has created two processes: *cmd.exe* and *tmp.exe* (at the blue links). By further clicking on the child process, we can retrieve the list of processes created by *cmd.exe* and *tmp.exe*. Then, the *cmd.exe* process has created two child processes: *reg.exe* and *timeout.exe*. The *tmp.exe* process did not create any child

¹MD5:b3eebe51ccc4a95815ddef3ef55604d2

process. The process networks of *cmd.exe*, *tmp.exe*, *reg.exe*, and *timeout.exe* are overlaid on top of the process timeline on request.

Trojan MultiInjector MultiInjector, under trojan classification, is a trojan that tries to inject code into other processes to hide or execute its payload and download and install other malware [132]. Figure 5.6(a) presents a sample of trojan MultiInjector under MalView analysis. MalView reveals the sequence of Process Create events generated by the malware and its interactions with other processes in the system, with multiple recurring patterns of function calls. Panel (b) shows the result of zooming into the most active/busy interval that was automatically detected by the tool, while panel (c) presents the outcome from filtering to highlight only interactions associated with the malware. The final result patterns are shown in panel (d).

By exploring details-on-demand via mousing over, as shown in panel (c), the first event in this sequence is Process Create from the malware to *cmd.exe* leading to the subsequent calls. Around 12:23:47, there are four consecutive Process Create calls from the malware to *net.exe*. The subsequent calls can be seen in panel (b) and panel (a) (for a broad view). Finally, the repeated event patterns associated with malware are clear in panel (d): one *Process Create* event from the malware to *cmd.exe*, followed by the four subsequences to *net.exe*. The behavior from this observation aligns with the characteristics of the malware of injecting code into other processes. The visualization helps to discern these low-level operations from the malware to other system processes.

5.4.2.4 Backdoor

A backdoor is a type of malware that provides unauthorized remote access to the compromised system by exploiting security vulnerabilities. The malware works in the background while hiding from the user. Meanwhile, it enables the attacker to have access to the victim's computer, such as databases and file servers, as well as running system-level commands. The process of injecting Backdoor is usually performed in two stages: First, a small file, called a dropper, is installed. Second, the dropper downloads the main malicious file from a remote location [98]. It is important to mention that Trojan and backdoor malwares are not the same: A Trojan might contain a backdoor, but a backdoor can execute as a stand-alone program without being a part of a Trojan.

The MalView visualization of malware Backdoor Androm execution on Windows 7 is presented in Figure 5.7. The accumulation of interactions presented in the top area chart divides the observation into two phases. The first phase heavily involves activities associated with the malware and *svchost.exe*. The last function call from the malware is

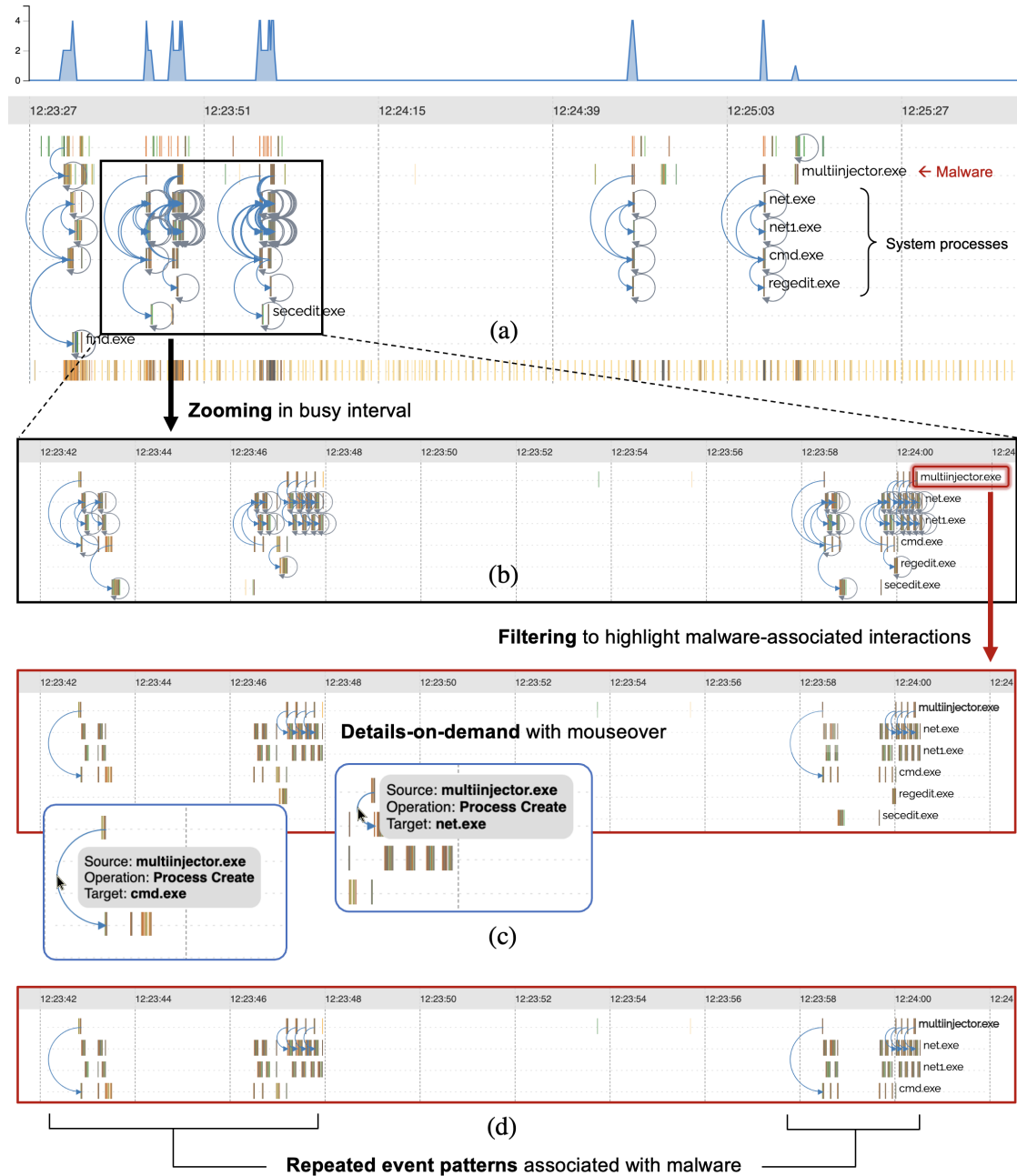


Figure 5.6: Trojan MultiInjector under MalView analysis: the sequence of Process Create events generated by the malware and its interactions with other processes in the system.

to *regasm.exe* (at the end of Box A), followed by an interesting recurring pattern in the second phase, as highlighted in Box B. This recurring pattern starts with a function call from *regasm.exe* itself to *schtasks.exe*, where the time between the two patterns is about two seconds. Here, process *regasm.exe* is the assembly registration tool, which reads

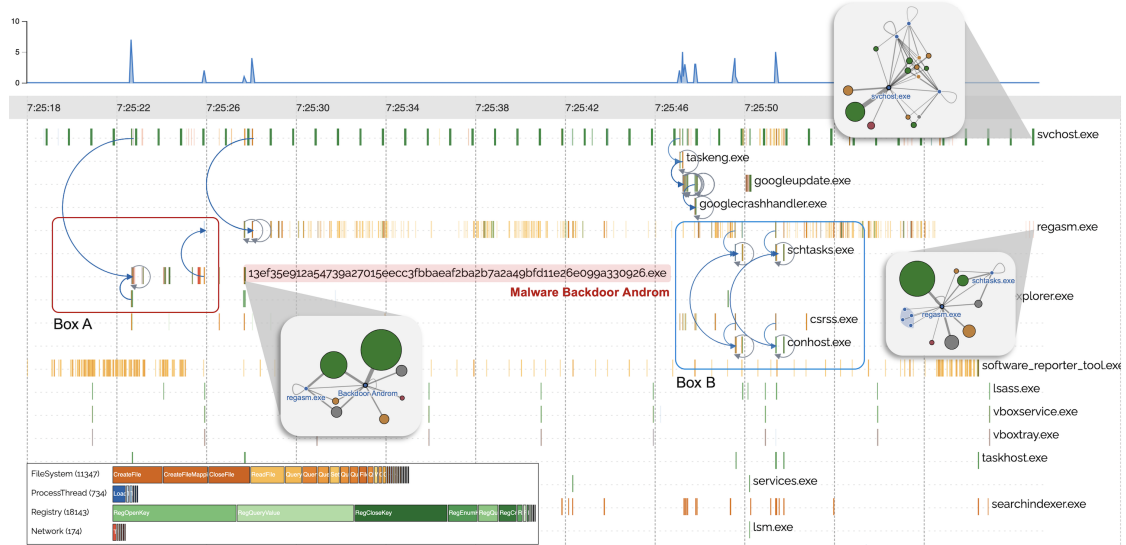


Figure 5.7: MalView visualization of malware Backdoor Androm on Windows 7. The top area chart demonstrates two separate phases: the first one with malware activities, ending with a call to *regasm.exe* (Box A) and the second with recurring patterns involving *regasm.exe* itself (Box B).

metadata within an assembly and adds necessary entries to the registry.

The overlay dependency graphs in Figure 5.7 open up several interesting findings. First, although its activities end early during the observation, the malware operates on multiple registry files, as shown by the large size of the green registry nodes. Second, *svchost.exe* has a long sequence of periodic operations on registry files and operates on numerous types of objects, as seen on the dependency graph. As the generic host process for Win32 services, it also makes function calls to the malware process twice, about five seconds apart. Finally, while *regasm.exe* interacts with five other processes, it only shares dependencies with *schtasks.exe*. The dependency graphs and process activity timeline are complementary and can effectively support the analytical reasoning of malware behaviors.

5.4.2.5 Ransomware

A typical ransomware program encrypts the victims' computer files and demands a ransom to restore access to the data. A ransomware program locks a system utilizes some visual messages, imposing law enforcement to threaten the target. The ransomware scam has matured over time, utilizing different methods to impair a computer. According to a report published by Symantec [167], the latest advancement prevents the computer from functioning and dismisses the client from gaining any access. The sys-

tem at such a stage displays a message that proclaims to be from a local law enforcement organization. The ransomware application asks for money in exchange for letting the users re-gain access to their systems. In recent news in July 2021 by Malwarebytes report [130], a severe ransomware attack was reportedly taking place against the popular Remote Monitoring and Management software tool Kaseya VSA. This attack has forced to immediately shut down the VSA servers, where Kaseya VSA was used to encrypt over 1,000 businesses. The attackers are asking for \$70M in exchange for a universal decryptor. Also reported by Malwarebytes [230], 35% of small and medium-sized businesses were under attack of ransomware. A lot of times, these organizations end up paying for the ransom. According to a multi-scan report from Virustotal, the sample studied in this chapter has a community score of 47 out of 72, i.e., out of 72 detection engines, 47 could identify it as a malicious executable.

Figure 5.8 shows the visualization for the dynamic activities of the ransomware *wannapeace.exe*². The malicious indicators presented by the MalView are as follows:

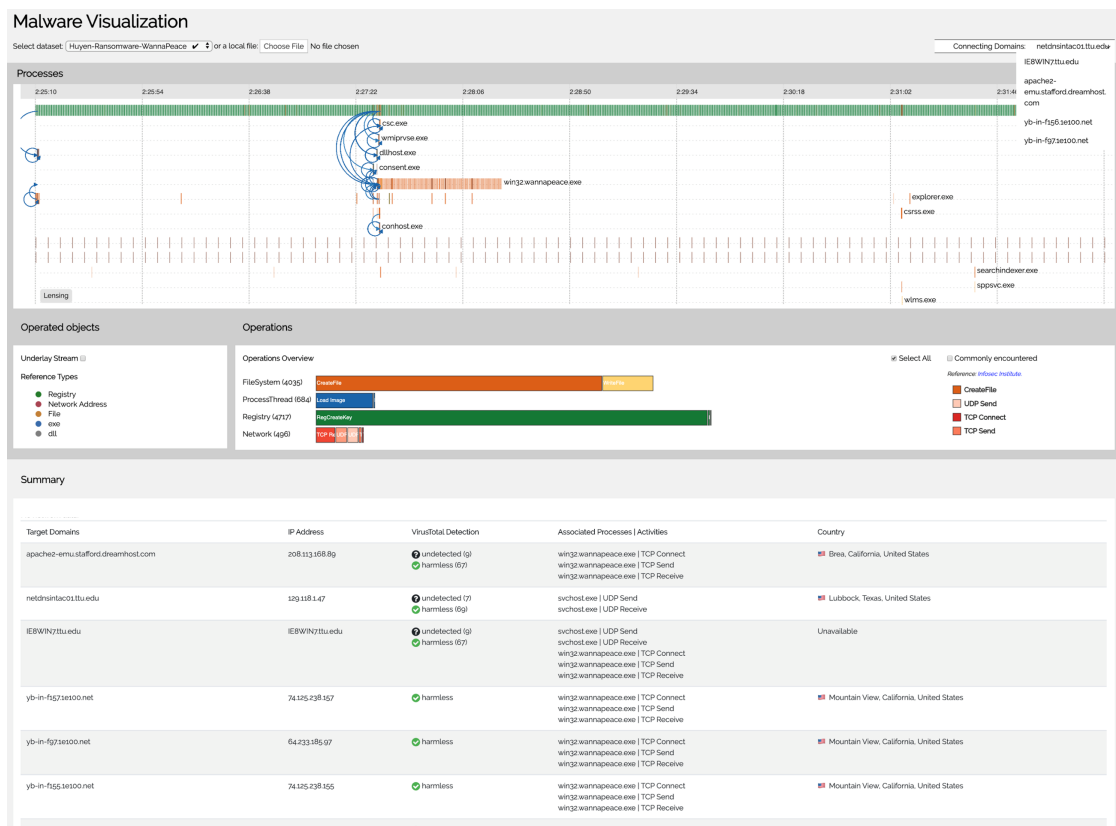


Figure 5.8: MalView analysis summary of ransomware WannaPeace on Windows 7.

²MD5: *ee6af98681d78b63f15d7e58934c6cc*

- Processes: The time window shows that the ransomware spawns: *conhost.exe* and *consent.exe*. The *conhost.exe* indicates that the ransomware is accessing the command line, whereas the *consent.exe* is an indicator that the program is trying to utilize the user access.
- Registry: It creates a registry key to make changes.
- Files: Malicious PE performs a lot of file operations. For instance, it performs 3434 times of *CreateFile* operations and 601 times of *WriteFile* operations.
- Network activity: The malware performs many TCP/UDP connection requests, sends, and receives.
- Domain Activity: It connects to seven different domains, as shown on the upper right corner of Figure 5.8.

5.4.2.6 Behavioral Malware

At the time of this writing, a search on Microsoft Security Intelligence threat search platform [135] returned 500 malware as Behavior type, in which the distribution of alert levels was as 400, 38, 3, and 16 for severe, high, moderate, and low, respectively. A behavior type of malware generally includes malware that exhibits suspicious activities, but it is not classified into a specific popular category of malware. This type of malware is difficult to detect because its activities can greatly vary depending on the intention of the underlying malware and the current user context. Our study of several malwares in “Behavior” type shows that these suspicious activities include 1) disabling system recovery, 2) deleting shadow copies, 3) hidden code executions, 4) creating files in the user’s system, 5) changing the registry key to run itself, and 6) accessing to *netsh.exe* to modify firewall configuration that allows itself to run on system startup. Examples of such behavioral malware include *Bladabindi.gen* [139], *Vawtrak.A* [142], and *Teerac.B* [141]. Furthermore, some behavioral malware (e.g., *MultiInjector* [140]) involves accessing the command prompt (CMD).

For instance, Figure 5.9 shows suspicious activities from an example of the Behavior malware type called *Bladabindi*. Panel (a) shows that it starts *netsh.exe* to modify firewall configuration to add itself as a permissible program. Panel (b) provides a piece of evidence as it sets the registry value (*RegSetValue*) on the user system to runs itself at Windows Startup for the same malware.

Program	bladabindi.exe	(a)
Operation	Process Create	
Event type	ProcessThread	
Timestamp	8:22:43.5055929 AM	
Path	c:\windows\system32\netsh.exe	
Detail	PID: 2032, Command line: netsh firewall add allowedprogram "C:\Users\IEUser\Desktop\Bladabindi.exe" "Bladabindi.exe" ENABLE	
PID	352	

Program	bladabindi.exe	(b)
Operation	RegSetValue	
Event type	Registry	
Timestamp	8:22:48.6046851 AM	
Path	hklm\software\microsoft\windows\currentversion\run\{84b68da87be0e442dacbf93621b7919}	
Detail	Type: REG_SZ, Length: 88, Data: "C:\Users\IEUser\Desktop\Bladabindi.exe" ..	
PID	352	

Figure 5.9: MalView shows suspicious activities from a Behavior malware, named *Bladabindi*. Panel (a) shows that it starts *netsh.exe* to modify firewall configuration. Panel (b) depicts that it sets the registry values to run itself.

5.4.2.7 Hacktool Malware

Hacktool is a piece of software that malicious attackers use to gain unauthorized access to user's devices [136]. As of the time of this writing, Microsoft lists 188 active entries as Hacktools, of which 93 are severe, 80 are high, and 15 are moderate in terms of alert levels [135]. The popular attacking channel for Hacktool is via insecure Universal Serial Bus (USB) communication design and Windows Autoplay features [176]. Malicious activities for Hacktool launched from USB include 1) changing registry settings, 2) installing a backdoor, 3) stealing confidential information, and 4) reading data encryption keys. Recently, besides Trojan, Hacktool is also the second most prevalent type of malware embedded in pirated software [118].

Figure 5.10 shows MalView view while analyzing a sample of Hacktool malware type named *Mailpassview* [143]. It first creates *svchost.exe* process (a). The *svchost.exe* process then creates *windows update.exe* (b). This process then creates several files like *holdermail.txt* (via using *vcb.exe*) to store "Browser Password Recovery Report," *pidloc.txt* to contain information of compromised computers (c). These are the pieces of evidence about the existence of Hawkeye Keylogger [181] to steal sensitive data (e.g., email password).

Determining whether the connecting domains from network activities are malicious or benign is important. The classification for malicious connecting domain for the malware *Mailpassview* is shown in Figure 5.11. Among the examined domains, *iplogger.com* is assessed as **malicious** and **suspicious** by VirusTotal, with the IP address



Figure 5.10: MalView view on a Hacktool malware type called *Mailpassview*. It first creates process *svchost.exe* (a), then *svchost.exe* starts *windows update.exe* (b), and then *windows update.exe* creates *pidloc.txt* (c)

Target Domains	IP Address	VirusTotal Detection	Associated Processes Activities	Country
iplogger.com	88.99.66.31	<div> malicious (1) suspicious (1) undetected (6) harmless (72) </div>	powershell.exe TCP Connect powershell.exe TCP Send powershell.exe TCP Receive powershell.exe TCP Disconnect	<div> Gunzenhausen, Bavaria, Germany </div>
deploy.static.akamaitechnologies.com	deploy.static.akamaitechnologies.com	<div> undetected (8) harmless (72) </div>	powershell.exe TCP Connect powershell.exe TCP Send powershell.exe TCP Receive powershell.exe TCP Disconnect	Unavailable
137-195-197-239.novotelecom.ru	37.195.197.239	<div> harmless </div>		<div> Novosibirsk, Novosibirsk Oblast, Russia </div>

Figure 5.11: The classification for malicious connecting domains from malware *Mailpassview*. The target domain *iplogger.com* is assessed as malicious and suspicious by VirusTotal. This domain is connected from *powershell.exe* via four activities: *TCP Connect*, *TCP Send*, *TCP Receive* and *TCP Disconnect*.

88.99.66.31 from Gunzenhausen, Bavaria, Germany. Recall from the chained calls shown in Figure 5.10: The process corresponding to the malware *mailpassview.exe* called and initiated *wscript.exe* with Process Create (panel “D” in Figure 5.10), then *wscript.exe* also called and initiated *powershell.exe* with Process Create. This chain continues with process *powershell.exe* connecting to malicious target domain *iplogger.com* with four different activities: *TCP Connect*, *TCP Send*, *TCP Receive* and *TCP Disconnect*. Here, *wscript.exe* is stored in C:\Windows\System32 and provides an environ-

ment in which users can execute scripts, which is different from the malicious programs that malware programmers or cyber criminals write and name it as *wscript.exe*.

5.5 Discussion

5.5.1 The Influence of Running Platforms on Malware Behavior

To examine how malware behaves in different platforms, we also executed multiple malware on Microsoft’s mainstream Windows OSs.

5.5.1.1 Email Flooder

We selected the “email flooder” malware to compare the visualization for this sample run in different platforms, including Windows XP, 7, and 10, as depicted in Figure 5.12. In particular, the output for Windows XP is simpler than the outputs produced by Windows 7 and 10. For example, the number of different processes for Windows XP is four vs. seven and nine for Windows 7 and 10, respectively.

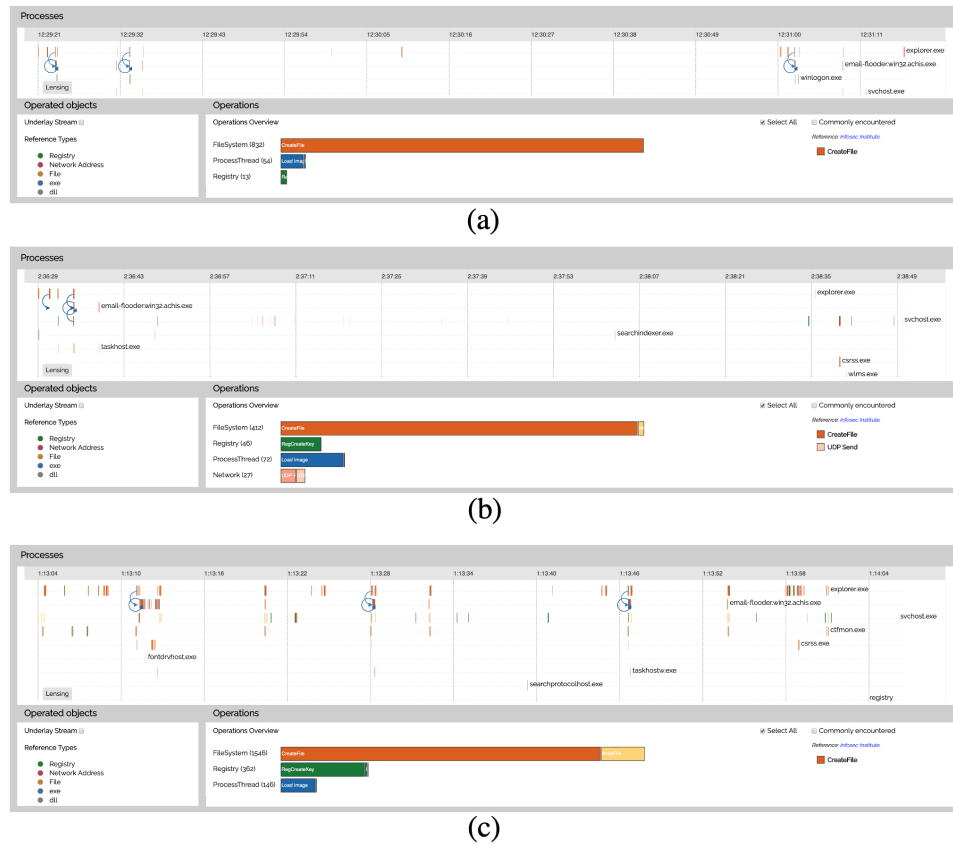


Figure 5.12: MalView visualizations of Email Flooder on (a) Windows XP, (b) Windows 7, and (c) Windows 10.

In addition, the total number of operations is much higher in Windows 10 than in Windows XP. It is observable that there is more information, including more processes, calls, dependencies, and activities in Windows 10 and 7 than XP. Since some of these pieces of information might be because of the Windows activities themselves and not the malware activities, tracking malware behavior in newer platforms might be more complicated.

5.5.1.2 Behavioral Malware

Figure 5.13 shows MalView views applied to *Bladabindi* malware executed on these three Windows operating systems in the top panel, middle panel, and bottom panel, respectively. In general, more platform-related processes are being executed in the latter two operating systems in comparison to Windows XP. However, its suspicious activities remain the same. In all platforms, it first starts *netsh.exe* to modify firewall configuration and then sets the registry values to run itself.

5.5.1.3 Patterns Across Platforms

One of the key features and benefits of employing visualization tools is to perform pattern detection and classification visually prior to delving into analytical approaches. MalView captures key features that are indicators for profiling classes or families of malware.

More specifically, using MalView it is possible to capture features such as volume of processes, registry activities, files manipulation and accesses, and network activities. As described below, these features are able to detect any “*behavioral*” patterns in the set of malware studied and thus enable us to classify them according to their dynamic behavior. Instead of trying to generate patterns of interest, in this study, we show how the analysis works based on malware behavior tracing, the kind of information it entails, and how the tool can enable analysts to quickly study the interaction of malware with system internals using selections, focus and context technique, and aggregations.

With MalView, we focus on the interactions of the malware program to other system internals processes. While Procmon, as the data provider, brings detailed information into each of the processes running in the system, the interval and log activity captured may be subjective to the person behind the capturing execution. To focus on the time interval in which we can witness the most significant amount of malware activity to other system internals processes, called *busy interval*, we applied focus and context visualization technique in MalView to support 1) close-up view for individual malware

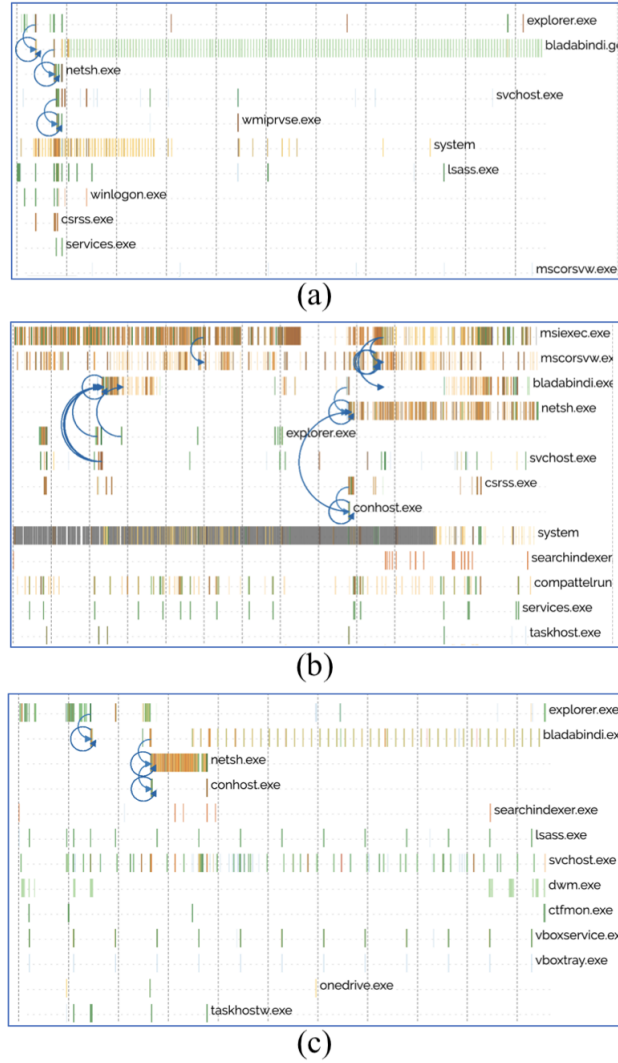


Figure 5.13: *Bladabindi* malware executions on different platforms: Windows XP (a), Windows 7 (b) and Windows 10 (c).

analysis and 2) standardization for malware comparison. To accommodate the context around the focal point, we select the interval that satisfies either ensuring the equal paddings to the first and last interaction to the boundary of the interval or equal paddings to the peak of the area chart - where there witness the highest amount of interactions.

Patterns of *Bladabindi* malware behavior across platforms: (a) Windows XP, (b) Windows 7, and (c) Windows 10, all under focus and context technique with busy interval length of 20 seconds, are shown Figure 5.14. By using mousing over an arc representing a function call, an user can observe the detailed information including type of operation, source and target processes. A recurring pattern observed from the *bladabindi* is the following sequence of calls: A *ProcessCreate* from *explorer.exe* to the malware, fol-

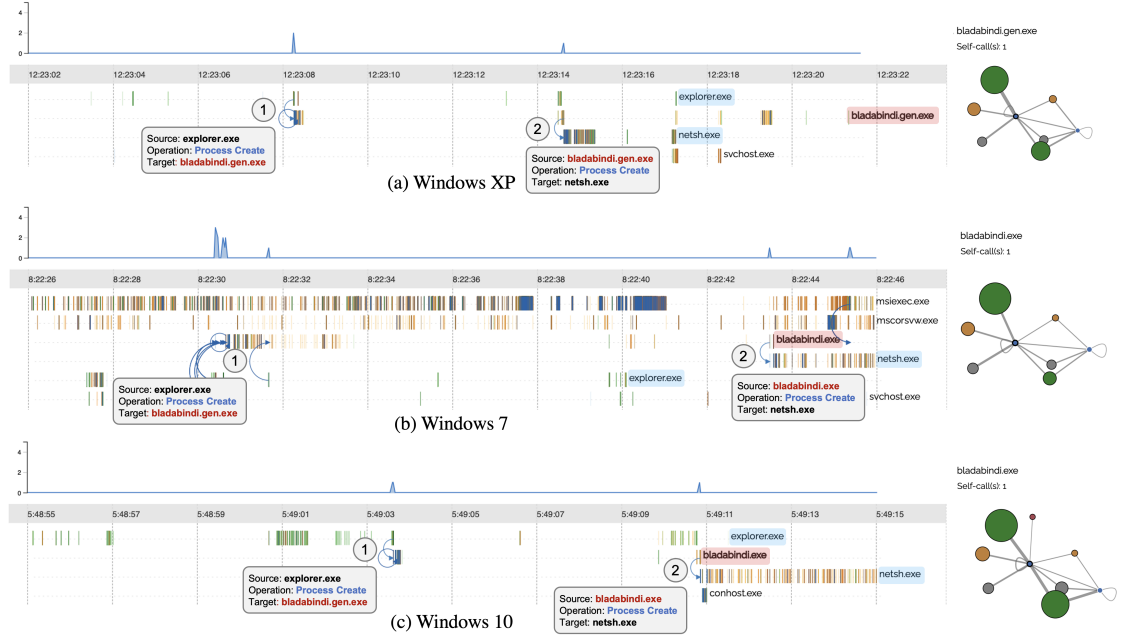


Figure 5.14: Patterns of malware behavior across platforms: *Bladabindi* malware executions on (a) Windows XP, (b) Windows 7, and (c) Windows 10, all under focus and context technique with busy interval length of 20 seconds. On the right end are the corresponding dependency graphs of the malware process. While different processes express different dependency graphs, as shown in Figure 5.5 and Figure 5.7, the dependency graph of *Bladabindi* malware is relatively consistent across different platforms.

lowing by a *ProcessCreate* from the malware to *netsh.exe*. As shown in Figure 5.5 and Figure 5.7, different processes produce very different dependency connections in terms of topology, grouping and volume. However, as presented on the right of Figure 5.14, the dependencies of the three *bladabindi* malware processes across different platforms demonstrate many similarities: the three biggest nodes that have the degree of one are all from registry (green), file (sand color), and dll (grey). For nodes with a degree of two - having connections with both *bladabindi* and *netsh.exe*, their categories are the same regardless of the running platforms. For further analysis, these patterns can serve as indicators for such classes of malware. Once again, we encounter the insights offered by the combination of timeline-based view and network view, providing us with complementary information.

5.5.2 Malware Visualization Tools vs. MalView

This section compares the features offered by MalView with the ones offered by some other malware visualization tools, including Hybrid [191] and AnyRun [14]. First, we

briefly review each visualization tool and then compare its features.

5.5.2.1 AnyRun: Interactive Online Malware Sandbox

Funded in 2016 by a Russian security researcher, Alexey Lapshin, AnyRun [14] offers a free “*interactive*” sandbox tool for dynamic analysis of malware. The tool enables uploading a suspicious file and, in the meantime, interacting with the sandbox and thus with malware to trigger some functionalities or execute macros embedded into the uploaded file. AnyRun offers several key features, as follows:

- The tool’s main feature is the visualization of interactive graphs and tree structure for comprehending malware. The feature helps visually identify suspicious processes, determine the family of malicious activities and patterns, and highlight external files that are downloaded by the malware.
- It also enables content analysis of different types of suspicious and malware files, including PCAP files (i.e., network activity dump).
- The tool also performs network analysis with the goal of tagging suspicious events. It analyzes Hypertext Transfer Protocol (Secure) (HTTP(s)) requests and responses along with their headers
- The tool can be used as an educational and training tool to assist the security experts to understand the structure of attacks through Mitre Att&ck Mapping [145].
- It enables opening web addresses (URLs) in different browsers and therefore helps in URL analysis and, more importantly, phishing attacks using various mainstream and supported browsers.
- The tool generates a fine-format report for publication and sharing purposes. The professionally-looking report consists of supporting screenshots, Process Behavior Graphs, indicators of being malicious/suspicious, and many other components.

5.5.2.2 Hybrid: Automated Malware Analysis Service

Hybrid [191] is a free malware analysis tool that enables both static and dynamic analysis. It utilizes several analysis reports and sandbox tools, including Falcon Sandbox [43], a dynamic analysis framework. In addition to the dynamic analysis offered by Falcon Sandbox, Hybrid integrates some other anti-virus tools such as VirusTotal, OPSWAT Metadefender, SIEM systems, NSRL (i.e., white listing), TOR (e.g., avoiding external

IP fingerprinting), Phantom, Thug Honey Client (e.g., URL exploit analysis), and Suri-cata (ETOpen/ETPro rules). The tool provides several useful analysis features such as:

- Risk summary and verdict of being malicious or benign.
- A good number of malicious/suspicious indicators
- A large set of network rules for intrusion detection and network analysis
- Integration with YARA [8] for rule-based pattern matching-based malware detection
- Analysis of a wide variety of files including binary samples and PCAP files
- Analysis of URLs for detection of phishing attacks

5.5.2.3 A Feature-based Comparison

This section aims to highlight the key features of AnyRun [14] and Hybrid [191] in comparison with the features offered by MalView. The comparison is performed through the classification of features into 1) general features, 2) behavioral activities and dynamic analysis, 3) structure-based and static analysis, and 4) network-level analysis. Table 5.1 lists the features classified into these four groups.

General Features MalView offers not only comparatively similar features but also additional features that are unique to MalView. More precisely, the tool offers features such as 1) compliance with InfoSec classification with respect to malicious processors and indicators (Feature #4), and 2) simplification of visualization through filtering and focusing only a subset of processes for the analysis (Feature #5).

Behavioral Activities and Dynamic Analysis Features The features related to dynamic analysis are considerably diverse. As a result, each analysis tool offers its own set of unique features. Given the fact that MalView mostly visualizes the output of Procmon [138], it is primarily a dynamic analysis tool. Depending on how the underlying malware visualization tool is implemented, most of these tools are able to visualize the “basic” sets of dynamic data captured through Procmon or similar utilities. For instance, as Table 5.1 shows, most of the behavioral features are visualizable by these three tools.

The major and key feature that is unique to MalView is the exploration of “*time dependencies between processes*” (Features #15 and #16). The visualization of time and

Table 5.1: A feature-based comparison of Anyrun [14], Hybrid [191] and MalView

#	Features	AnyRun	Hybrid	MalView
I) General Features				
1	Integrated with VirusTotal [94]	*	✓	✓
2	Integrated with MetaDefender [168]	*	✓	
3	Integrated with CrowdStrike Falcon [43]		✓	
4	Compliance with InfoSec Classification [99, 100]			✓
5	Filtering (i.e., Simplification) Capabilities			✓
6	Classification into Malicious/Benign	✓	✓	✓
7	List of Imported DLLs	✓	✓	✓
8	Statistics of Extracted Executable Files	✓	✓	✓
9	Highlights Malicious Indicators (e.g., report from anti-viruses, installation activities, creation of a windows session/station, termination of a session, and spanning a lots of processes	✓	✓	✓
10	Highlights Suspicious Indicators (e.g., inspection of PE files, suspicious API calls, locating of resource files, deleting executable files, creating files in Windows directory, importing suspicious APIs)	✓	✓	✓
II) Behavioral Activities and Dynamic Analysis				
11	Behavioral Graphs/Activities	✓		✓
12	Different Classes and details of Processes			✓
13	Call graph and Number of Calls for each Process/category	✓		✓
14	Scalar representation of process calls			✓
15	Dependencies among processes/executable files			✓
16	Time Dependencies between processes/executable files			✓
17	Lists dangerous/suspicious system level activities (creating/changing important files: registry, SVCHOST to execute hidden code, writes to start menu, etc.)	✓	✓	
18	Registry/File Activities	✓	✓	✓
19	List of Registry/File changes and modifications and events	✓	✓	✓
20	Statistics about events and registry activity including read/write/ delete events			✓
21	Statistics about executable files	✓	✓	✓
22	Provide Falcon Sandbox Report (Dynamic Execution)	✓	✓	
23	Basic Risk Assessment (e.g., access to clipboard, spanning processes, reading the computer name, and injecting into explorer)	✓	✓	
24	Extract memory strings and tokens	✓	✓	
25	The name of the extracted executable files	✓	✓	✓
26	List the size of different Sections of PE files	✓	✓	Partially
III) Network Level Analysis				
27	Domain Name System (DNS) Requests	✓	✓	✓
28	Host Connections	✓	✓	✓
29	HTTP Traffic/Requests	✓	✓	✓
30	Network Threats	✓		✓
31	PCAP Download	✓		
32	Registry Changes	✓		
33	Details on GET/POST methods	✓		

*: The feature is under development.

process dependencies are an important part of malware analysis in order to comprehend the nature of the underlying malware.

Signature-based and Static Analysis As stated earlier, MalView is primarily a visual analytics tool based on the output of the dynamic analysis of the underlying application or malware. As a result, it is less focused on visualizing static features of executable

files. However, MalView is integrated with several static analysis tools, including VirusTotal, and thus is capable of capturing this information and visualize them accordingly. VirusTotal is able to capture static information such as the size of header files, type of files, PE Specific, and other static and signature-based features. As a result, MalView can visualize all the information captured by VirusTotal and uses its API to retrieve this information and visualize them accordingly.

Network Level Analysis Similar to signature-based and static analysis features, MalView is less focused on visualizing purely network-level features. However, given the strength of Procmon in capturing all related processes and events, MalView is capable of visualizing the network-level events and processes captured by Procmon and thus provides a process-level view on this network-level information.

5.6 Conclusion

This chapter introduces MalView, an interactive visualization platform for hybrid analysis and diagnosis of malware. Our approach first represents the behavioral properties of the major malware classes (such as Trojan or backdoor), aiming to capture the common visual signatures of these malicious applications. MalView implements a web-based prototype for demonstrating our approach to analyzing 60 malware samples from seven different classes. The behavior aspects of these malware files are captured using Process Monitor (i.e., Procmon [138]) on three different platforms (Windows XP, Windows 7, and Windows 10). The functionality and features offered by MalView are designed and developed based on a thorough literature review and a comparison with the state-of-the-art malware analysis tools, including AnyRun and Hybrid. In order to have better insight regarding the features offered by MalView, a feature table is presented in which MalView is compared with AnyRun and Hybrid analysis tools. The feature comparison is performed based on four classes of features. The feature table demonstrates that MalView comparatively implements most of the features offered by the other two tools. In addition, the time and processed dependencies, the key features of MalView, are implemented in the prototype, making the analysis more thorough. Given the ability to process, visualize and analyze the system activities and put them into a comprehensive view, MalView can serve as an informative and potential interest to developers, engineers, and practitioners outside the laboratory.

There are several lines of research that can be explored through visual analytics when complemented by conventional static and dynamic analysis: The early detection of zero-day vulnerability and malware is a grand challenge. There are several machine learning-

based approaches for addressing this problem [3, 122]. With the capability of visual analytics facilitating explainable machine learning [54, 48], applying visual analytics techniques to detecting and analyzing unknown and zero-day malware is an interesting research approach that can be explored using MalView. The key feature of MalView is its features in demonstrating time and process dependencies that occurred during static and dynamic analysis. A potential research direction is to model malware behavior through recurrent neural networks on the visual signatures and then predict malware behaviors or even classify suspicious programs into a particular class of malware. It would also be interesting to model malware samples through genome alignments and then model the malware classification or detection problem through deoxyribonucleic acid (DNA) or sequence matching approaches. The sequence matching might be useful in capturing the core malicious functionalities of obfuscated malware. The obfuscation techniques employed by the obfuscating tools often follow similar patterns, and thus we expect the control-flow graphs produced for all these obfuscated malicious applications share fully or partially the same core. MalView will offer a visual analytic approach to spot these similar patterns in the execution traces. Once a section of the underlying execution trace is identified as obfuscated, it can be ignored by the user of MalView and then enables the users to focus on other parts of the malware in order to comprehend it. A second approach would be to employ existing de-obfuscated tools to de-obfuscate the malware under investigation (MUI) and then let Procmon generates the de-obfuscated traces of execution and processes.

Acknowledgment

This work was supported by the National Science Foundation (NSF) under Grant No: 1821560.

CHAPTER 6

AUTOMATED EVENT DETECTION FRAMEWORK

The insights offered by the timeline-based view in conjunction with the network view in previous chapters inspired us to explore a more abstract concept of this integration. In this chapter, we delve into the implication of this combination to facilitate event detection. The visualization of events supports time-series data exploration and analysis, especially for discerning significant changes. Although events can be detected by visual inspection, automated analysis techniques can help facilitate gaining insights and reducing cognitive load. This study proposes an automated event detection framework that advocates for 1) utilizing the potential underlying graph structure within time-series data to explore hidden relationships and 2) decreasing cognitive load associated with visual inspection by introducing automatic detection of events. The system that adopts this framework allows users to investigate the data as the first step in the exploratory data analysis process. We demonstrate the feasibility of this framework with the implementation on an existing visualization platform and examine how well the framework help improve the task of event detection in these systems. Several specific use cases that motivated this direction can be found under Sections 4.4.2.2, 5.4.2.4, and 5.5.1.3, in the preceding chapters.

6.1 Introduction

An event, while it may appear to be a straightforward concept at first glance, opens up a plethora of intriguing discourse in visual data exploration and analysis. Depending on how the data was captured and how the analyst views it, the occurrence of an event can be inherent, or in the case of a compound event: it can be described as a construct emerging from the data. The visualization of events supports data exploration and analysis, explanation, and presentation that assist data interpretability, monitoring, and decision-making process. Events can be detected by visual inspection of a time graph [11]: patterns over time can be easily identified, including sudden changes or fluctuations that may indicate the occurrence of an event in data. However, visual inspection alone may not guarantee to be sufficient for detecting events in all cases, as some events may be subtle or involve hidden attributes, and hence require more advanced statistical techniques for detection. In dealing with information overload problems, visual analytics approaches present automated analysis techniques combined with

interactive visualization to foster effective understanding and reasoning of data [111]. Automated techniques facilitate data exploration by providing efficient methods to help identify patterns, features, and anomalies, reducing the cognitive load associated with manual visual inspection.

The study of underlying relationships among entities is a valuable asset for understanding the hidden structure of data. Network visualization and analysis present a powerful approach to showcase and help understand the interconnectivity among different entities in a graph structure. The use of network visual representation proves effective in community detection and pattern discovery [166]. However, compared to network representation, timeline-based visualizations are still the majority in visual representations to display temporal event sequence data [87]. The lack of network analysis integration into event detection in time-series data leads to missed opportunities in exploration, analysis, and presentation.

This study proposes an automated event detection framework. This framework advocates for 1) utilizing the potential underlying graph structure within time-series data to reveal hidden attributes, and 2) decreasing cognitive load associated with visual inspection by introducing automatic detection of events. On the sequential time-series side, we apply statistical techniques for detection. On the network side, we first run relationship mining, followed by identifying the properties and temporal features that fit the exploratory tasks. These relationships can be inherent (e.g., interacting accounts on social media) or require further mining techniques (e.g., shared resources among processes in a system). The detection candidates, or significant changes, found in the sequential time-series side and network side, are then combined and presented to the users for verification. We then implemented this framework on top of existing visualizations to examine and compare the benefits the approach brings.

The inspiration for this work stems from lessons learned from previous projects on temporal events with timeline-based and network visualizations [157, 156]. While visual inspection serves effectively to a certain extent, automatic event detection would be extremely helpful in discerning patterns beyond the directly observable. The study is also inspired by previous literature on temporal events [12, 84] and network evolution analysis by Trier [206] and Ahn et al. [4]. The flow of this framework is based on the sense-making loop for visual analytics by Keim et al. [111] and the simple model of visualization by van Wijk [211].

This chapter introduces an automated event detection framework for time-series data to utilize the potential underlying graph structure to explore hidden relationships. The system that adopts this framework can help users discern significant changes in data

during the exploratory data analysis process. In addition, we demonstrate the use of the framework to augment existing visualization applications and examine the extent to which it helps improve event detection tasks. We first give an overview of the background and prior literature related to this study in Section 6.2. Section 6.3 presents the main methodology of the framework, including design guidelines for subsequent applications. Section 6.4 demonstrates the case studies of application onto visualizations. Section 6.5 concludes the chapter and presents an outlook for future work.

6.2 Related Work

Automated analysis techniques have been introduced in visualization to help gain insights for optimizing and navigating complicated processes [111, 32]. In fact, fully-automated analysis only work reliably for *well-defined* and *well-understood* problems [111]; therefore, it is crucial to provide the means to combine the strengths of humans and machines that leverage their respective distinct capabilities, with visualization becoming the medium. The framework will reflect this perspective regarding the user-specified inputs and defaults.

Established visualization and visual analysis methodologies harness the linear, temporal aspect of time-series data, resulting in the prevalence of timeline-based visualization and sequence analysis, demonstrated from a recent survey by Guo et al. [87]. Previous work has addressed the visualization of events in time-series [116, 16, 219, 67, 73, 129] in terms of the design for visual encodings, interactions, and along the text stream data type. To detect such events, the Page-Hinckley algorithm introduces a sequential analysis technique to monitoring changes in signal processing [76] and was adopted in visual analytics for event detection in spatial time series [11] and temporal social networks [204]. The test considers a cumulative variable m_T , defined as the accumulated difference between observed values and their current mean. At each step, the minimum value of m_T is also computed as $M_T = \min(m_t, t = 1 \dots T)$. The Page-Hinckley test monitors the difference between m_T and M_T , and fire an alarm when $m_T - M_T > \lambda$. The value of the threshold λ is set by the user, in which increasing λ will reduce the number of false alarms, but it may also result in overlooked or postponed changes.

From a network point of view, different network formations can have different structures, but there are certain universal topological characteristics and relationships shared among network entities, called *structural properties*. Table 6.1 presents the structural properties of the network that are commonly encountered in visualization and visual analytics from previous studies. The descriptions in this table are extracted from previous

Table 6.1: Structural properties of a network structure commonly encountered in visualization and visual analytics systems.

Source: Trier (2008) [206]; Monge and Contractor (2003) [146]; Wasserman and Faust (1994) [220].

Structural property	Description	References
Network size	Number of nodes in a network, e.g., active users.	Arikan & Dalton [15], Trier [206].
Density	Ratio of actual pairwise connections between n nodes of a network to the total possible relationships among these n nodes.	Yi et al. [228], Arikan & Dalton [15].
Degree (in-degree vs. out-degree)	Number of direct links with other nodes. Out-degree measures the relationship events initiated by the observed node, in-degree measures the events initiated by neighbors adjacent to the observed node.	Chen & Morris [36], Ye et al. [227], Bastian et al. [21].
Connectedness	Extent to which each pair of nodes is joined by some path, either direct or indirect.	Dang et al. [57], Broek et al. [62].
Direction	Extent to which link is pointed from one node to another.	Kang et al. [109], Durant et al. [68]
Betweenness Centrality	Number of shortest paths between pairs of nodes that pass through the observed node, e.g., the role could be a person who forwards important messages in an email exchange network.	Gloor & Zhao [78], Perer & Shneiderman [174], Gloor et al. [77].
Degree Centrality	Simple centrality measure, counting the relative share of contacts of a node.	Mutton [152], Shamma et al. [196], Yi et al. [228].
Reciprocity (Symmetry)	Extent to which relationship is bidirectional. If there is a connection from node A to node B and vice versa, then this relationship is called reciprocal.	Moody et al. [148], Monge & Contractor [146]
Group Membership	Collection of nodes belonging to a group. The membership may stem from a natural arrangement (e.g., animal herds), or be developed during analysis (e.g., by clustering)	Kang et al. [109], Bremm et al. [27], Morris et al. [149].

work by Trier [206], Monge and Contractor [146], and Wasserman and Faust [220].

Ahn, Plaisant, and Shneiderman [4] present a task taxonomy of temporal network visualization and suggest the temporal features less explored to guide future design opportunities. The study emphasizes the tasks in temporal visualization analysis of graphs. This chapter addresses a complementary view: leveraging the network to better explore time-series data, combining human input towards well-defined problems and automated analysis. Specific to the temporal aspect of network evolution, Table 6.2 gives an overview of temporal features based on the findings of Ahn et al. [4]. Recent works indicate that the temporal aspect and ordering of interaction that can introduce higher-

Table 6.2: Overview of Temporal features in network evolution (Ahn et al. [4]).

Individual Events	
Single Occurrences	Atomic temporal events, e.g., the addition or deletion of an entity, e.g., a node.
Replacement	Simultaneous deletion and addition, e.g., edge direction change.
Birth or Death	The start and end point of a life cycle of a group or larger.
Aggregated Events: Shape of Changes	
Growth or Contraction	Increases or decreases of an entity property over time, e.g., number of node/link additions per month (structural property) or average number of messages per user per month (domain property).
Convergence or Divergence	Growth or contraction of a property following by a stable period, or a stable property becoming unstable, e.g., convergence point of the transitivity (structural property).
Stability	No or little change over time.
Repetition	Repetition of specific patterns over time, e.g., repeated communications between two users.
Peak or Valley	Extent to which a property increases or decreases abruptly and then returns to its earlier value, e.g., sudden peak within a time range.
Aggregated Events: Rate of Changes	
Speed: Fast or Slow	Amount of change in a given time period, e.g., speed of growth of nodes within a month.
Acceleration or Deceleration	Rate of change of speed, e.g., identifying whether a change is getting faster or slower.

order, non-dyadic dependencies are not captured by state-of-the-art graph models, as indicated by Hackl et al. [88] and Lambiotte et al. [119].

From time-series data, the essential first step involves data transformation to extract the relationships among data entities. Through mining mutual social tags, previously developed systems have the ability to extract such relationships to construct the network of active users [4]. Perer et al. [173] presented a set of systems that mine, aggregate, and make inferences on a social graph from social media data. In the domain of cybersecurity, for example, internal processes in a system can share operated objects such as registry and dynamic-link libraries. By identifying the shared resources and the interaction graph among processes, including the malware [155], we can discern the irregularities in behavior between the malicious and benign entries.

6.3 Methodology

The first and foremost aim of this framework is to guide users' attention to where there are potential events – significant changes in the data. An overview of the proposed automated event detection framework is presented in Figure 6.1. The framework is created to serve the data exploration stage, where the analyst wants to perform initial investigations of the data. This approach is based on (1) data transformations to derive one or more network structures from sequential time-series data, (2) automated analysis with the foundation on well-defined definitions of what constitutes an event, (3) user-specified parameters such as a threshold to consider a worthy event and the granularity under which the data is aggregated, and (4) interactive visual elements to aid exploration and further analysis. The flow of exploratory analysis with event detection starts with data transformation to extract the relationships among data entities. To perform the initial analysis, default values are set for event formulation (including event shapes and temporal network features), parameters, and thresholds, but the users should be able to change these values to examine different alternatives.

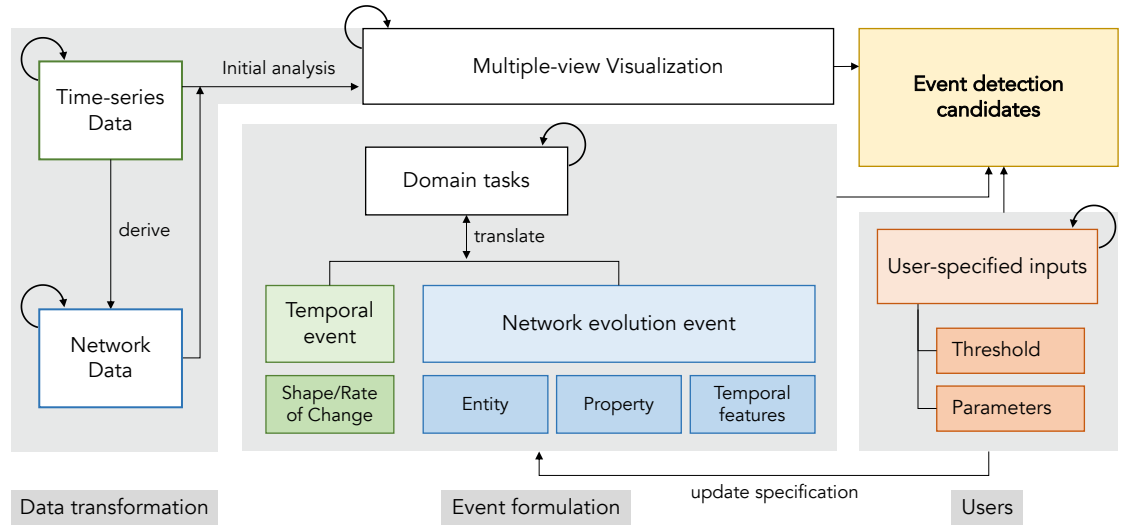


Figure 6.1: Overview of the proposed automated event detection framework.

6.3.1 Data Transformation

The beginning data transformation stage involves deriving the network structures from sequential time-series data, as shown in the leftmost segment in Figure 6.1. This can be achieved by first identifying the entities of interest that fit the requirements of the domain problem. Then the analyst collates these entities with data entries from the original

time-series form to define their characteristics. To formulate a network, the next step is to extract the connections among entities. Based on how network connections should be represented from the problem description, these links can be achieved in various forms, including extracting communication patterns, mining mutual social tags, and grouping shared resources among the entities. The looping arrows in Figure 6.1 denote the refining process for each data format. The initial analysis is applied to the dual data structures to construct the proper visual representation. With two data types, multiple-view visualization is encouraged to provide comprehensive views on multiple facets of data representation.

6.3.2 Time-series Event Formulation

The formulation of an *event* varies in specification based on the perspective we view the data. Time-series data presents an ordered sequence of time-value pairs, where they are often but not always at equidistant time intervals [151]. Events in time series are defined at an atomic level: as timestamped data points or measurements [116, 16] that can occur anytime. A common case of time-series data occurs in a temporal event sequence data set, where an event is the main object to be captured. In this case, the event is inherent as a building block, and the data set contains one or more sequences constituted of such events [87]. However, when time series incorporates other data types such as text stream, an event is no longer an intrinsic element but rather a complex construct: unique pattern or irregular activities in comparison with average signs across the temporal dimension [219], or an occurrence causing a change in the volume of text data characterized by topic and time and associated with people and location [67].

An example of a complex temporal event is described as follows. In the aftermath of an earthquake, as discussed in Chapter 4, the sewer lines suffered breakage, urging citizens to use social media platforms to seek attention by directing their messages towards the Water Utilities Department and other influential people within the community for assistance and spreading the call for help. This complex event includes the earthquake as a main event, broken sewer lines as a subsequent event with specific locations, people in the community, related influencers or authority figures, along with the timestamps where these events occur.

In order to determine the measures for temporal characteristics, we first identify the possible shapes an event can exhibit. We summarize the event shapes from previous work [11, 12, 84] and present in Figure 6.2. The common denominator shared among previous research is that an event signifies a significant change in data, where the degree of significance is often demonstrated by an abrupt surge or slump followed by a

return to the prior value. Gregory and Shneiderman [84] described classes of shape for generic time-series analysis, including line, spike, sink, rise, drop, plateau, valley, and gap shapes. Andrienko et al. [12] presented methods for finding moments of interest by periodicity and peak detections.

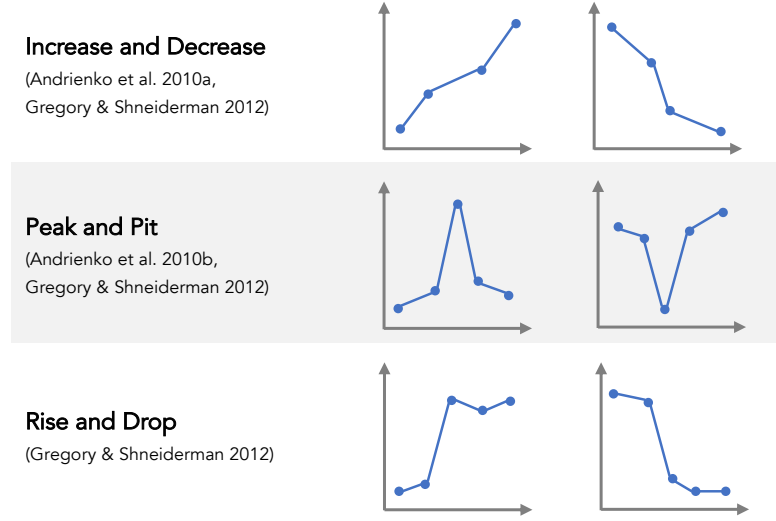


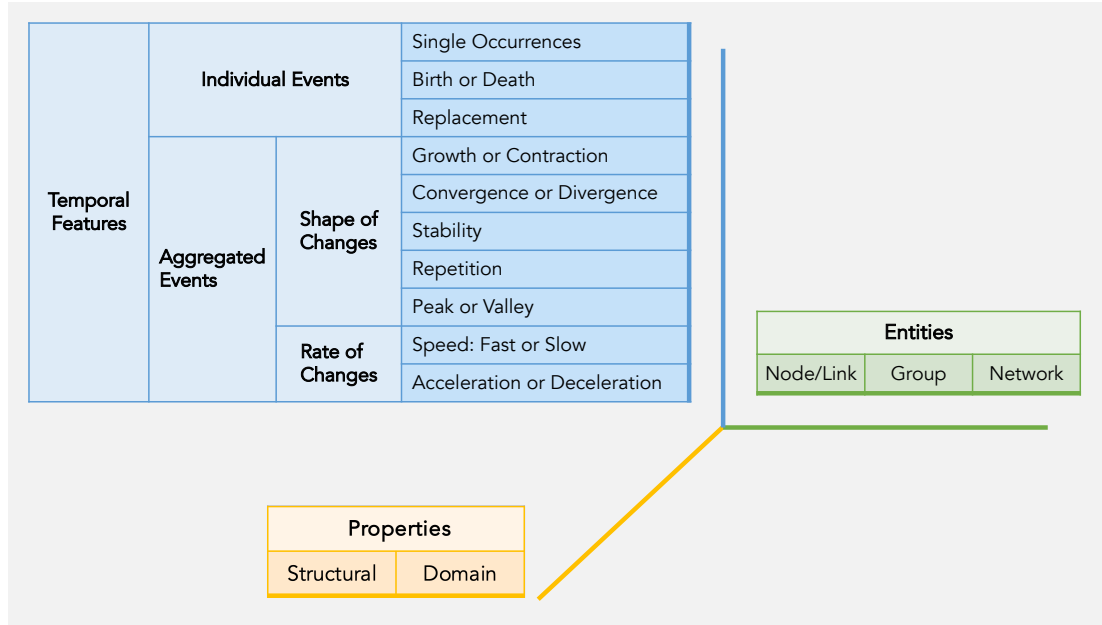
Figure 6.2: Common shapes encountered as events in time-series analysis.

- **Increase and Decrease**, as depicted in the top row of Figure 6.2. The line shape of increase and decrease are useful to describe consistent behaviors of growing or declining.
- **Peak and Pit**, also known as Spike and Sink. These shapes depict a temporal behavior where a variable abruptly changes in one direction, followed by a significant reverse shift, as shown in the middle row of Figure 6.2.
- **Rise and Drop**, referring to the sustained change in the mean value over time. These shapes are decided by two stable periods and one change period in between, as depicted in the bottom row of Figure 6.2.

6.3.3 Network Event Formulation

We use three dimensions to view a network structure and its evolution over time [4]: entity, property, and temporal feature. Their interplay is shown in Figure 6.3.

- The *entities* are the objects of interest: node/link, group, or the network itself. This dimension is shown in green in Figure 6.3.



Example:

Temporal Feature:
Growth

Structural Property:
Betweenness Centrality

Entity: Node

→ The growth of betweenness centrality of a node in a network, or:

The growth over time of the influence a node has over the flow of information in a graph.

Figure 6.3: The space of network evolution tasks in three dimensions: Entities, Properties, and Temporal Features. Each point in this space represents an element in the 3-fold Cartesian product from the three dimensions. For example, with Temporal feature as *Growth*, Structural property as *Betweenness Centrality*, and Entity as *Node*, we have the task of examining the growth of betweenness centrality of a node in a network. This task can then be translated as the growth over time of the influence a node has over the flow of information in a graph.

- The characteristics of these entities are their *properties* (shown in yellow in Figure 6.3). The structure of network data leads to two different types of properties: Structural properties and domain properties. The former refers to the topological relationships among network entities, and the latter defines the related information specific to that domain problem and independent of network structure. Table 6.1 gives an overview of the structural properties of the network that are commonly encountered in visualization and visual analytics.
- The last dimension involves *temporal features*: how we observe, identify, and make comparisons between the entities and their properties over time (shown in blue). Table 6.2 provides an overview of temporal features in network evolution.

Note that the aggregated events with the shape of changes in Table 6.2 refer to a similar categorization of temporal shapes as in Figure 6.2, where they share the characterization of significant temporal changes.

The above three dimensions provide a design space to explore the analysis-related tasks of network evolution. Each point in this space represents an element in the corresponding 3-fold Cartesian product. For example, with Temporal feature as *Growth*, Structural property as *Betweenness Centrality*, and Entity as *Node*, the task can be seen as examining the growth of betweenness centrality of a node in a network. The description of betweenness centrality can be referenced in Table 6.1. This task can then be translated in a more descriptive language as the growth over time of the influence a node has over the flow of information in a graph.

6.3.4 User-specified Inputs

Although the automated part of this framework is executed with defaults, the user can specify certain inputs and parameters to explore alternative results. The user-specified inputs are categorized as follows:

Granularity The scale under which data is aggregated. Finer granularity means fewer data points are grouped in each time interval. While there are common ways to choose time intervals to group data, such as by second, minute, or hour, granularity can also be defined by the number of intervals the analyst wants to divide the entire timeline into. Besides the timeline, granularity can divide other universal units, such as the total amount of exchanges.

Threshold The magnitude that must be exceeded for a shape to be considered an event. The greater this threshold, the greater the chance to reduce the number of false positive events, but it also may lead to overlooked ones.

Parameter(s) The variable factor(s) forming the definition of an event, e.g., the number of consecutive points that construct a peak, such as 3-point or 5-point peaks [84]. An example of a 1-point peak can be found in the middle row of Figure 6.2.

6.3.5 Design Guidelines

We identify the following design guidelines to facilitate the design process for systems that adopt the framework. These guidelines help the design process in the two middle

levels of visualization design [150]: *operation and data type abstraction* stage, where we address the mappings between domain problem descriptions to abstract operations and data types, and *visual encoding and interaction design* stage, where we take the previous stage’s output as input and specify the visual encoding decisions.

G1: Extract potential relationship connections from time-series. First, we identify the entities of interest, such as user accounts or shared resources, and the meaning a connection between two distinct entities should convey. Second, we transform the data from time series to graph structure(s), which may contain multiple “snapshots” of network structure, depending on the time window size or the granularity in which the data should be aggregated.

G2: Enable user interactions to formulate the detection and enhance exploration. Systems that adopt this framework should allow users to provide input that can be used to refine the system’s detection results. At the same time, the interaction capabilities help users navigate through the data in a more intuitive and exploratory way, potentially revealing deeper insights.

G3: Leverage multiple views to simultaneously present linear time attribute and network relationships. Multiple views can utilize the strengths of multiple visual encodings; therefore, the two facets of linear time-series view and network evolution can be represented distinctly. Moreover, when combined with interactivity, multiple views will allow for in-depth data exploration at multiple levels of detail.

G4: Foster transparency in presenting detection results. The rationale that leads to results should be presented in a way that allows users to understand why such findings are demonstrated and what would be the alternate results if they adjust the thresholds and other parameters. The transparency in reasoning helps enhance trustworthiness in automatic detection results inspired by previous studies [186, 117].

6.4 Results

In this section, we present the results of applying the framework to two previous case studies with timeline-based and network visualizations in the preceding chapters. We discuss the results on EQSA (introduced in Chapter 4) first, followed by the outcome from the application on MalView (introduced in Chapter 5). In terms of visual encodings for detections, we use colored bounding boxes, inspired by the common practice used in object detection [184] and previous event detection work [129].

6.4.1 Case Study with EQSA

We implemented the framework to augment the visualization presented in the Earthquake Situational Analytics (EQSA) dashboard, from [157] and Chapter 4. This dashboard presents an interactive exploratory tool for earthquake situational analytics using social media. The data is from a benchmark dataset by VAST Challenge 2019: Mini-Challenge 3. EQSA is designed to facilitate characterizing the condition across the area around the earthquake zone, with support to discern related events, resources to be allocated, and responses from the community.

In this use case, we examine two coordinated views: timeline-based and network visualizations, as depicted in Figure 6.4. The timeline-based view shows the volume of messages generated over time, and the network view illustrates the connections among user accounts within the community. To showcase the volume of these connections, we selected the degree centrality of the network as the main focus.

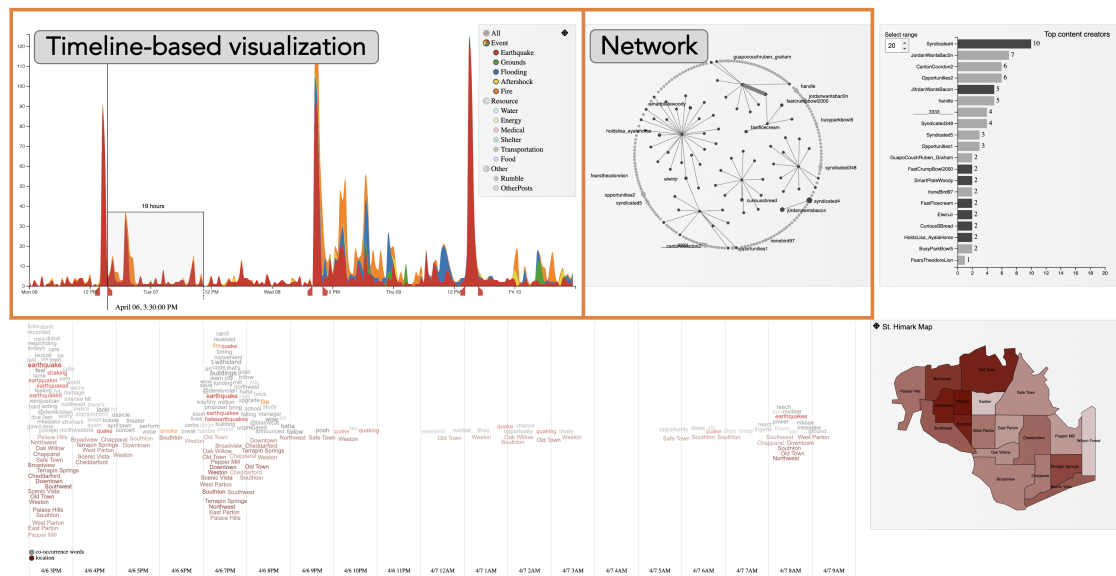


Figure 6.4: Original visual interface of the Earthquake Situational Analytics [157]. We will focus on the two highlighted panels for Timeline-based visualization and Network.

In the language of three dimensions of network evolution analysis [4]:

- The observed entity here is the entire network captured by the time frame.
- The property is degree centrality, a structural property that represents the number of immediate contacts of a node.
- The temporal features here are growth as the shape of changes and the underexplored features of speed and acceleration as the rate of changes.

A series of demonstrations for the usage of the event detection framework in EQSA are shown from Figure 6.5 to Figure 6.9.

Figure 6.5 demonstrates the temporal peak detection with $granularity_{spike} = 13$ and $threshold_{ratio} = 6$. Boxes 1, 2, 4, and 5 are then verified and kept for further exploration. $granularity_{spike} = 13$ indicates that the timeline is divided into 13 equal intervals. This number is the result of scanning with different granularity and selecting the granularity with the highest accuracy. In this case, the peaks are 1-point peaks, which are determined by:

$$V_p > \lambda * \frac{\sum_{i=1}^{p-1} V_i + \sum_{i=p+1}^N V_i}{N - 1}$$

Here, p is a peak if its value V_p is greater than the product of the ratio threshold λ and the mean of the remaining values within the time frame. λ is adjustable by the user, and lower λ will increase the number of detections, including false positives. In Figure 6.5, $\lambda = 6$.

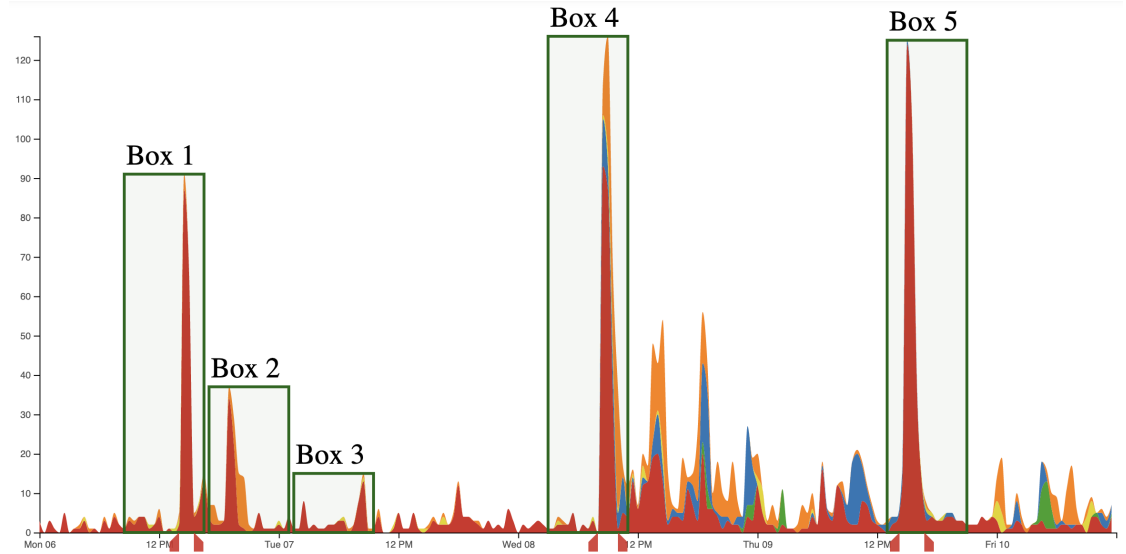


Figure 6.5: Temporal peak detection with $granularity_{spike} = 13$ and $threshold_{ratio} = 6$. Boxes 1, 2, 4, and 5 are verified and kept for further exploration.

The top 5 most significant network growth/contraction for the structural property of centrality at $granularity_{growth} = 20$ is shown in Figure 6.6. The largest value is 1.2, significantly greater than other values, showing the biggest growth at the corresponding granularity. In this case, $granularity_{growth}$ signifies the value by which the total volume of messages is divided, leading to different widths of bounding boxes in Figure 6.6. Figure 6.7 shows details into the largest growth of 1.2: After the first peak found in Box

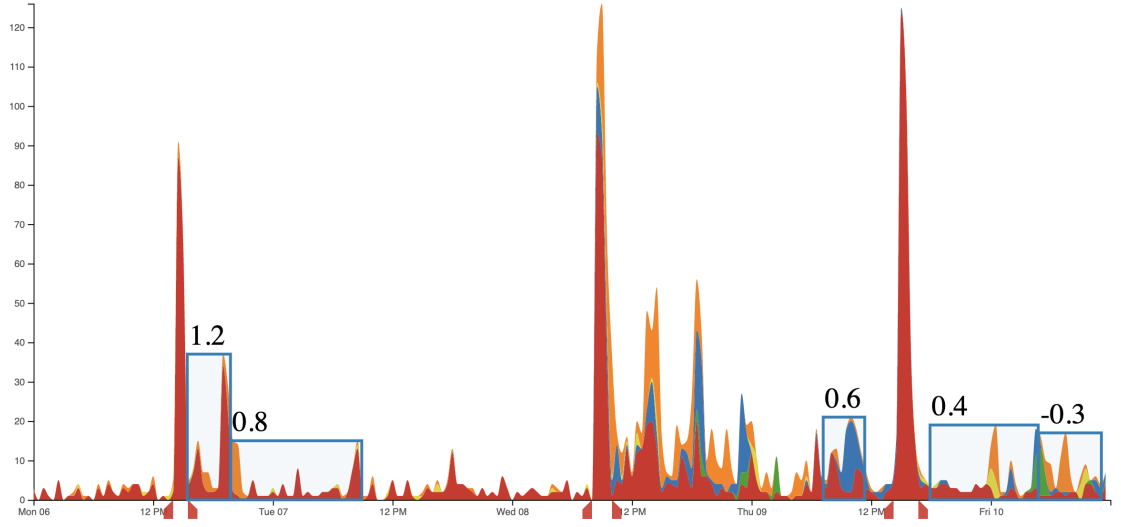


Figure 6.6: Top 5 biggest network growth of centrality at $granularity_{growth} = 20$.

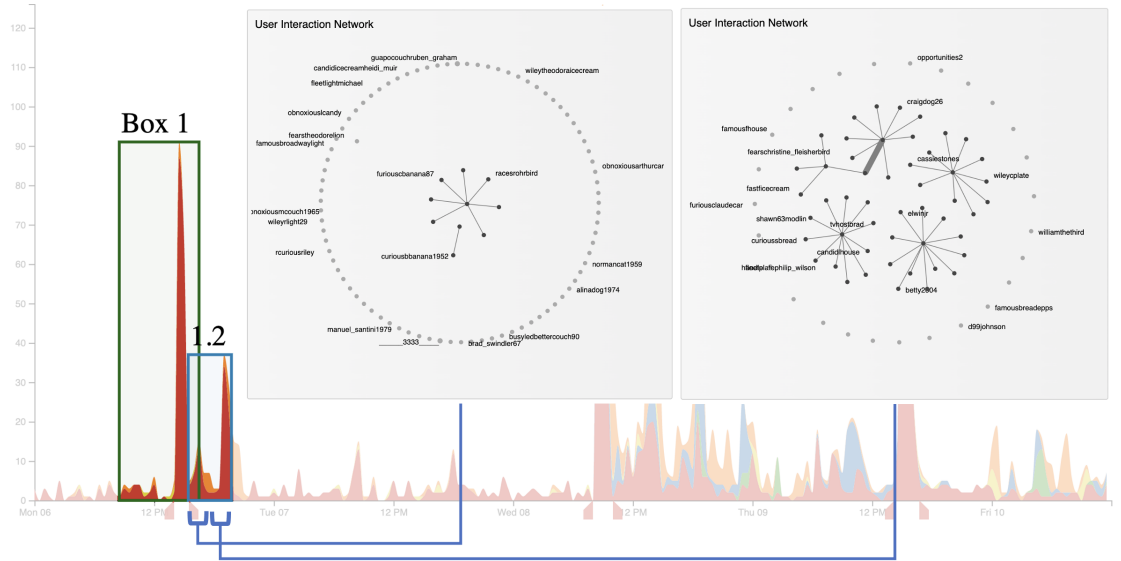


Figure 6.7: Details into the largest growth of 1.2: After the first peak found in Box 1, the network was scarce but then grew significantly and formed four major clusters, hence the increase in centrality.

1, the connections in the network were scarce but then grew significantly and formed four major clusters, hence the sharp increase in centrality.

The highest speeds of changes were detected at $granularity_{speed} = 17$, as shown in Figure 6.8. Negative signs indicate that the amounts of changes decrease: the zoomed-in interval depicts a decline to zero centrality, as shown in the detailed network panels.

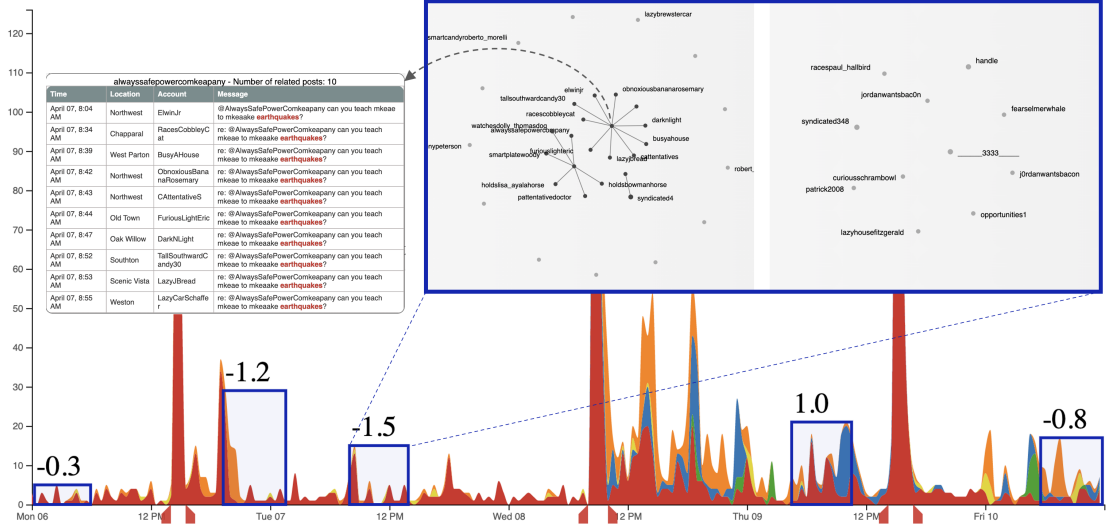


Figure 6.8: Highest speeds of changes were detected at $granularity_{speed} = 17$. Negative signs mean the amounts of changes decline; e.g., the zoomed-in interval shows a significant decrease to zero of centrality.

Figure 6.9 demonstrates the top 5 of the corresponding acceleration of centrality of the network at $granularity_{speed} = 17$. Since acceleration is the derivative of velocity (speed), the same $granularity_{speed}$ is used here. The detection of the event corresponding to the acceleration of -1.0 has a surge in the volume of messages at the second half of the interval but with a deceleration: the new connections grow slower.

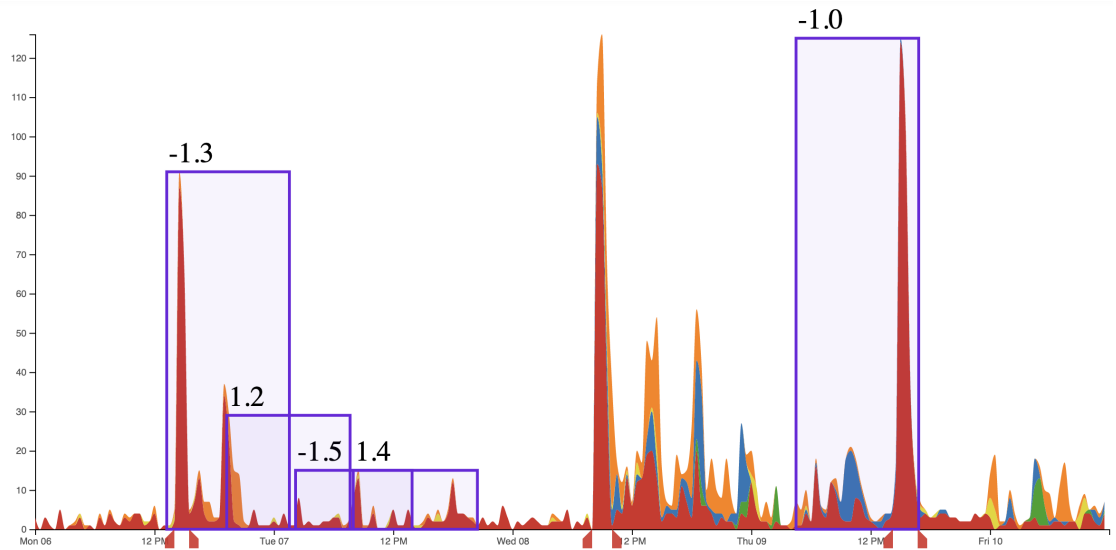


Figure 6.9: Top 5 of the corresponding acceleration of centrality of the network at the same $granularity_{speed} = 17$.

6.4.2 Case Study with MalView

As previously described in Chapter 5, the network visualization in MalView demonstrates the dependencies between the malware process with the objects it operates on and shared resources with other processes in the system. In other words, we use the 1.5-diameter network: we consider the observed node's connections and all the connections among those nodes. An example of a dependency graph with malware *bladabindi.gen* [139] is shown in Figure 6.10. Note that only the *exe* nodes can actively create connections. Here, the only other process the malware connects to is *netsh.exe*; therefore, we consider the connections of the malware nodes and all the connections among these nodes with *netsh.exe*.

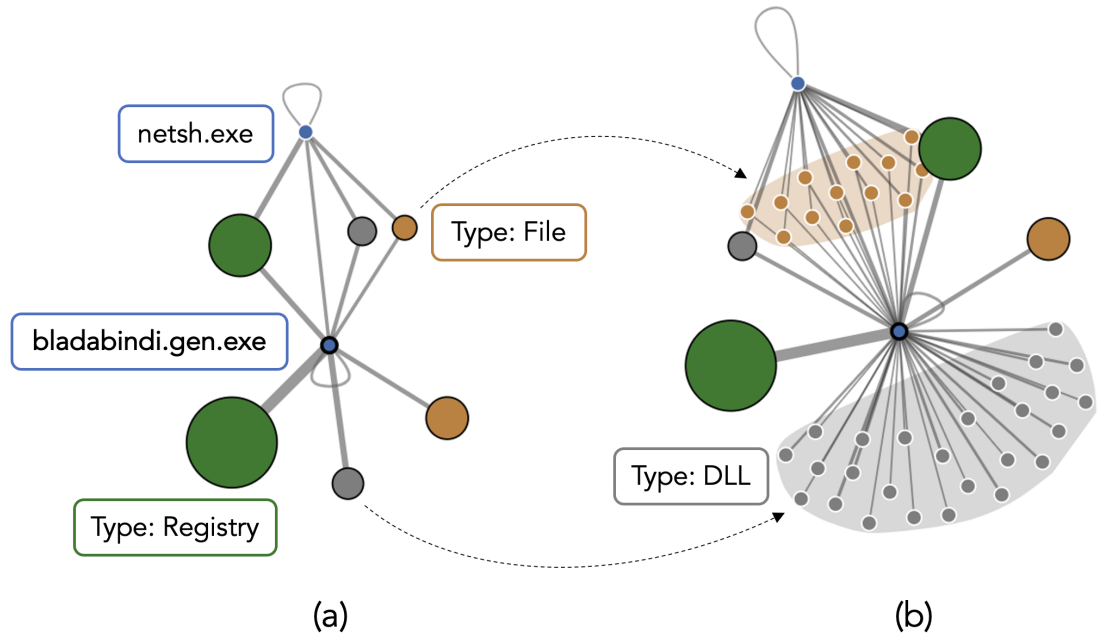


Figure 6.10: An example of dependency graph with with malware *bladabindi.gen* and its shared resources with *netsh.exe*. Figure (a) shows all nodes in the compact form and Figure (b) shows the expanded forms of two corresponding nodes of type File and DLL in (a). On the expanded view (b), edges are depicted explicitly.

We examine the growth and stability of edge count from the dependency graph of the malware process, therefore locate the time frames where the malware exploits the largest number of resources in the system. In the language of three dimensions of network evolution analysis:

- The observed entity here is the network corresponding to the malware.
- The property is a structural property of edge count.

- The temporal features here are growth and stability over time.

Figure 6.11 shows the detection results with malware *bladabindi.gen*. We measure the growth of connections that the malware created per second. Two time points were identified with a significant change in the growth of edge creation, marked with red boxes as A and B (top panel). They corresponded to the two peaks of the lower panel at 12:23:08 and 12:23:20. The system was captured at 12:23:02, but the malware did not make any activity until 12:23:08. In that first second, the malware created 725 connections to other resources in the system, the highest number in the whole capture. The edge creation declined, followed by a surge at 12:23:30 of 314 connections. After the initial high fluctuation period, the activity fell into a stable period, with the lowest edge creation per second of 8 and the highest of 11. Two corresponding network views are shown to the right of each peak, demonstrating that the malware accessed registry objects (in green) the most at both times. The timeline-based view alone would not be able to discern this information.

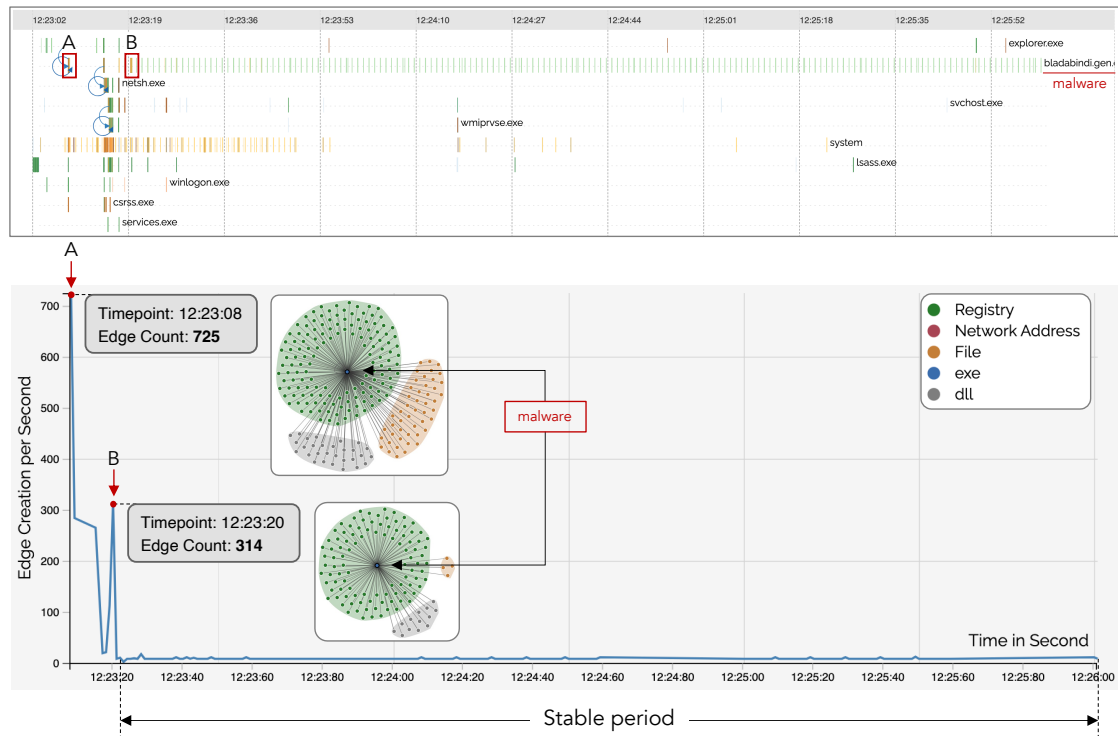


Figure 6.11: Detection results with malware *bladabindi.gen*. The upper panel shows the process activity view of MalView, and the lower panel shows a detailed view of the growth of edge creation per second. The two peaks are detected, marked as A and B, with their corresponding dependency graphs.

6.5 Conclusion

This chapter presents an automated event detection framework for time-series data. In the era of big data, timely and effective extraction of meaningful events is crucial. Given this, our framework leverages network structure to reveal hidden attributes and relationships, along with time-series characteristics, to detect events automatically. We explore various shapes of temporal changes and three dimensions associated with network evolution events: entity, property, and temporal features. Our event detection framework has been implemented on top of existing visualization platforms, acting as an extended layer that offers enhanced event detection capabilities. The findings suggest that the added layer helps discern the events that are not immediately observable before, simplifies the comprehension of complex time-series data, and makes extracting meaningful insights less of a daunting task. Future work involves further development to enhance user interaction and improve detection accuracy. We aim to design more intuitive and responsive interfaces, enabling users to navigate, manipulate, and interpret the visualized data more effectively.

CHAPTER 7

CONCLUSION

This dissertation contributes novel visualization designs and techniques that foster interactive systems for exploratory analysis of time-series data, advancing the practicality of visualization in comprehending both qualitative and quantitative data. These concepts are implemented in tools and libraries, including five visualization platforms and one framework for data analysis and communication. This chapter will review these contributions, their limitations, and present outlook for future work.

7.1 Review of Research Contributions and Impacts

From the qualitative perspective, this dissertation pioneers the use of WordStream, a visualization tool for topic evolution in text data. The design of WordStream prioritizes context, meaningful word frequency, and fluctuations over time, enabling the representation of text data in an aesthetically appealing and comprehensible manner. This practice becomes particularly relevant when tackling the progression of dynamic topics over time, especially when applying to rich, context-laden qualitative text data.

The utility of WordStream was then extended by WordStream Maker, a user-friendly development that increased accessibility to WordStream visualization creation, enabling individuals without programming skills to utilize its capabilities. The usefulness and adaptability of WordStream were demonstrated through two different case studies, focusing on social media analytics and educational assessments. These findings lay the groundwork for further exploration of WordStream’s integration into different scenarios, such as longitudinal text analysis studies.

In the field of quantitative visualization, this dissertation presents MalView, a comprehensive and interactive platform specifically designed to comprehend the complexities of malware behavior. The development of MalView addresses an existing gap in the field, enhancing malware analysis effectiveness by providing visual signatures of diverse malware classes. The usefulness and practicality of MalView were demonstrated through comparative case studies, highlighting its potential as a valuable tool for cyber security experts.

This dissertation then integrates the lessons learned from developing the qualitative and quantitative data visualization dashboards to cultivate the creation of an Automated

Event Detection Framework. This framework augments the manual visual data inspection process by automating event detection, a practical improvement given the increasing complexity of time-series data. The application of the framework to the previously developed qualitative and quantitative dashboards emphasized its utility and validated its effectiveness.

In terms of impacts, this dissertation establishes a foundation for further research at the intersection of visualization and time-series data, specifically focusing on domain problems with both qualitative and quantitative data. By introducing new techniques and tools, the study suggests new solutions for researchers and practitioners, particularly in the realm of qualitative research. These developments support the advancement of domain-specific visualizations, extending from social media analytics to malware analysis. Moreover, the creation of an automated event detection framework shows promise for future automated data analysis. It offers solutions to minimize the cognitive load associated with manual visual inspection and elicits various considerations for data transformation.

7.2 Limitations and Future Directions

Besides the contributions of this dissertation, it is important to acknowledge several limitations, which pave the way for valuable future research opportunities.

Usability for non-experts: While WordStream Maker was developed to support users without programming knowledge to create WordStream visualizations, a learning curve might still be involved. The tool’s user-friendliness for non-experts may be hindered by some inherent complexities in calculating the visualization and may require further simplification or user tutorials/training documents. We have received and resolved the issues raised by users on GitHub and recognized that showing explicit error notifications to tell users what is going on could help improve user experience with the tool.

Transferability/Generalizability of the case studies: While the case studies demonstrated the applicability of WordStream and MalView in certain contexts, different domains may have unique requirements and challenges that could limit the utility of these tools. One potential solution to this issue is conducting comprehensive user studies to gather insights into the perceived usefulness, practicality, and potential improvements. Moreover, user study provides an opportunity to detect any unforeseen issues that may emerge only under specific use cases.

Sensitivity to input data quality: The quality and reliability of the results generated by the developed tools and framework are contingent on the quality of the input data. For

example, the output of MalView relies on how the user captured the process monitoring data, and it could be altered by the internet connection or obfuscation techniques used by the malware to hide the activity. Another example is the trade-off between the quality of NLP and the need for a lightweight model. Noise, outliers, or inaccuracies in the original data could potentially lead to misleading or incorrect visualizations and event detections.

7.3 Concluding Remarks

Through the development of interactive tools such as WordStream, WordStream Maker, and MalView, the research expands our capabilities to navigate and understand both qualitative and quantitative time-series data effectively. The journey of this dissertation has helped bridge the existing gaps in specific domain problems and marked the potential of visualization in new territories and applications. We are encouraged to pursue the development of WordStream, partly because many colleagues and end-users have reached out to us regarding the tool, sharing with us how the tool helped them solve their problems and encouraging us with new research opportunities and further collaborations. Moving forward, it is my aspiration that these tools will continue to evolve, empowering individuals to discover new insights and deepen their understanding of their data.

BIBLIOGRAPHY

- [1] ABOALELA, R., AND KHAN, J. I. Visualizing concept space of course content. In *2015 IEEE 7th International Conference on Engineering Education (ICEED)* (2015), pp. 160–165.
- [2] ABRAMSON, C. M., AND DOHAN, D. Beyond text: Using arrays to represent and analyze ethnographic data. *Sociological methodology* 45, 1 (2015), 272–319.
- [3] ABRI, F., SIAMI-NAMINI, S., KHANGHAH, M. A., SOLTANI, F. M., AND NAMIN, A. S. Can machine/deep learning classifiers detect zero-day malware with high accuracy? In *IEEE BigData* (2019).
- [4] AHN, J.-W., PLAISANT, C., AND SHNEIDERMAN, B. A task taxonomy for network evolution analysis. *IEEE Transactions on Visualization and Computer Graphics* 20, 3 (2014), 365–376.
- [5] AIGNER, W., MIKSCH, S., SCHUMANN, H., AND TOMINSKI, C. *Visualization of time-oriented data*, vol. 4. Springer, 2011.
- [6] ALAEIYAN, M., PARSA, S., AND CONTI, M. Analysis and classification of context-based malware behavior. *Computer Communications* 136 (2019), 76–90.
- [7] ALJARRAH, E., ELREHAIL, H., AND AABABNEH, B. E-voting in jordan: Assessing readiness and developing a system. *Computers in Human Behavior* 63 (2016), 860–867.
- [8] ALVAREZ, V. M. Yara: The pattern matching swiss knife for malware researchers. <https://virustotal.github.io/yara/>, Accessed 2019.
- [9] AMAR, R., EAGAN, J., AND STASKO, J. Low-level components of analytic activity in information visualization. In *Information Visualization, 2005. INFOVIS 2005. IEEE Symposium on* (2005), IEEE, pp. 111–117.
- [10] ANDERSON, B., STORLIE, C., AND LANE, T. Improving malware classification: Bridging the static/dynamic gap. In *Proceedings of the 5th ACM Workshop on Security and Artificial Intelligence* (New York, NY, USA, 2012), ACM, pp. 3–14.
- [11] ANDRIENKO, G., ANDRIENKO, N., MLADENOV, M., MOCK, M., AND POELITZ, C. Extracting events from spatial time series. In *2010 14th International Conference Information Visualisation* (2010a), pp. 48–53.
- [12] ANDRIENKO, G., ANDRIENKO, N., MLADENOV, M., MOCK, M., AND PÖLITZ, C. Discovering bits of place histories from people’s activity traces. In *2010 IEEE Symposium on Visual Analytics Science and Technology* (2010b), pp. 59–66.

- [13] ANGELINI, M., ANIELLO, L., LENTI, S., SANTUCCI, G., AND UCCI, D. The goods, the bads and the uglies: Supporting decisions in malware detection through visual analytics. In *2017 IEEE Symposium on Visualization for Cyber Security (VizSec)* (2017), IEEE, pp. 1–8.
- [14] ANY.RUN LLC. ANY.RUN - interactive online malware sandbox. <https://any.run/>, Accessed 2019.
- [15] ARIKAN, B., AND DALTON, B. Micro fashion network: Color. <https://plw.media.mit.edu/people/arikan/2005/microfashionnetwork/>, 2005.
- [16] ARIS, A., SHNEIDERMAN, B., PLAISANT, C., SHMUELI, G., AND JANK, W. Representing unevenly-spaced time series data for visualization and interactive exploration. In *Human-Computer Interaction - INTERACT 2005* (Berlin, Heidelberg, 2005), M. F. Costabile and F. Paternò, Eds., Springer Berlin Heidelberg, pp. 835–846.
- [17] AZZAM, T., EVERGREEN, S., GERMUTH, A. A., AND KISTLER, S. J. Data visualization and evaluation. *New Directions for Evaluation 2013*, 139 (2013), 7–32.
- [18] BACA, H. A. H., LUZ PALOMINO VALDIVIA, F. D., ATENCIO, Y. P., IBARRA, M. J., CRUZ, M. A., AND BACA, M. E. H. Covidstream: Interactive visualization of emotions evolution associated with covid-19. In *Annual International Conference on Information Management and Big Data* (2020), Springer, pp. 540–551.
- [19] BARTH, L., KOBOUROV, S. G., AND PUPYREV, S. Experimental comparison of semantic word clouds. In *Experimental Algorithms* (Cham, 2014), J. Gudmundsson and J. Katajainen, Eds., Springer International Publishing, pp. 247–258.
- [20] BASOLE, R. C., SEUSS, C. D., AND ROUSE, W. B. It innovation adoption by enterprises: Knowledge discovery through text analytics. *Decision Support Systems 54*, 2 (2013), 1044–1054.
- [21] BASTIAN, M., HEYMANN, S., AND JACOMY, M. Gephi: an open source software for exploring and manipulating networks. In *Proceedings of the international AAAI conference on web and social media* (2009), vol. 3, pp. 361–362.
- [22] BERNARD, J., HUTTER, M., REINEMUTH, H., PFEIFER, H., BORS, C., AND KOHLHAMMER, J. Visual-interactive preprocessing of multivariate time series data. In *Computer Graphics Forum* (2019), vol. 38, Wiley Online Library, pp. 401–412.
- [23] BEST, D. M., BOHN, S., LOVE, D., WYNNE, A., AND PIKE, W. A. Real-time visualization of network behaviors for situational awareness. In *Proceedings of the Seventh International Symposium on Visualization for Cyber Security* (New York, NY, USA, 2010), VizSec '10, Association for Computing Machinery, p. 79–90.

- [24] BÖRNER, K., BUECKLE, A., AND GINDA, M. Data visualization literacy: Definitions, conceptual frameworks, exercises, and assessments. *Proceedings of the National Academy of Sciences* 116, 6 (2019), 1857–1864.
- [25] BOSTOCK, M. Towards Reusable Charts. <https://bost.ocks.org/mike/chart/>, 2012. [Online; accessed August 4, 2022].
- [26] BOSTOCK, M., OGIEVETSKY, V., AND HEER, J. D³ data-driven documents. *IEEE transactions on visualization and computer graphics* 17, 12 (2011), 2301–2309.
- [27] BREMM, S., ANDRIENKO, G., ANDRIENKO, N., SCHRECK, T., AND LANDESBERGER, T. V. Interactive analysis of object group changes over time. In *EuroVA 2011* (2011).
- [28] BYRON, L., AND WATTENBERG, M. Stacked Graphs - Geometry & Aesthetics. *IEEE Transactions on Visualization and Computer Graphics* 14, 6 (2008), 1245–1252.
- [29] CALLAHAN, S. P., FREIRE, J., SANTOS, E., SCHEIDEGGER, C. E., SILVA, C. T., AND VO, H. T. Vistrails: Visualization meets data management. In *Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data* (New York, NY, USA, 2006), SIGMOD '06, Association for Computing Machinery, p. 745–747.
- [30] CAPPERS, B. C., MEESSEN, P. N., ETALLE, S., AND VAN WIJK, J. J. Eventpad: Rapid malware analysis and reverse engineering using visual analytics. In *2018 IEEE Symposium on Visualization for Cyber Security (VizSec)* (2018), IEEE, pp. 1–8.
- [31] CASEY, W., HAVRILLA, J., HINES, C., METCALF, L., AND SHELMIRE, A. Sparse Representation Modeling for Software Corpora. *Results of SEI Line-Funded Exploratory New Starts Projects* (2012), 43.
- [32] CENEDA, D., ARLEO, A., GSCHWANDTNER, T., AND MIKSCH, S. Show me your face: Towards an automated method to provide timely guidance in visual analytics. *IEEE Transactions on Visualization and Computer Graphics* 28, 12 (2022), 4570–4581.
- [33] CHARIKAR, M. S. Similarity estimation techniques from rounding algorithms. In *Proceedings of the thirty-fourth annual ACM symposium on Theory of computing* (2002), ACM, pp. 380–388.
- [34] CHARY, M., PARIKH, S., MANINI, A. F., BOYER, E. W., AND RADEOS, M. A review of natural language processing in medical education. *Western Journal of Emergency Medicine* 20, 1 (2019), 78.

- [35] CHECK POINT SOFTWARE TECHNOLOGIES LTD. October 2018's most wanted malware: For the first time, remote access trojan reaches top 10 threats. <https://blog.checkpoint.com/2018/11/13/october-2018s-most-wanted-malware-for-the-first-time-remote-access-trojan-reaches-top-threats-cryptomining/>, Accessed 2019.
- [36] CHEN, C., AND MORRIS, S. Visualizing evolving networks: Minimum spanning trees versus pathfinder networks. In *IEEE symposium on information visualization 2003 (IEEE Cat. No. 03TH8714)* (2003), IEEE, pp. 67–74.
- [37] CHEN, S., LI, S., XIE, L., ZHONG, Y., HAN, Y., AND YUAN, X. Earthquakeaware: Visual analytics for understanding human impacts of earthquakes from social media data. In *2019 IEEE Conference on Visual Analytics Science and Technology (VAST)* (2019), IEEE, pp. 122–123.
- [38] CHI, M. T., LIN, S. S., CHEN, S. Y., LIN, C. H., AND LEE, T. Y. Morphable Word Clouds for Time-Varying Text Data Visualization. *IEEE Transactions on Visualization and Computer Graphics* 21, 12 (2015), 1415–1426.
- [39] COLLINS, C., VIÉGAS, F. B., AND WATTENBERG, M. Parallel tag clouds to explore and analyze faceted text corpora. *Proceedings of the VAST 09 - IEEE Symposium on Visual Analytics Science and Technology* (2009), 91–98.
- [40] CONTI, G., DEAN, E., SINDA, M., AND SANGSTER, B. Visual reverse engineering of binary and data files. In *Visualization for Computer Security* (Berlin, Heidelberg, 2008), J. R. Goodall, G. Conti, and K.-L. Ma, Eds., Springer Berlin Heidelberg, pp. 1–17.
- [41] COOK, K., FALLON, J., AND CROUSER, J. Mini-Challenge 3: Voice from the People, 2019.
- [42] CORVUS FORENSICS. Virus share. <https://virusshare.com/>, Accessed 2023-07-30.
- [43] CROWDSTRIKE. Automated malware analysis & sandbox: Falcon sandbox. <https://www.crowdstrike.com/>, Accessed 2019.
- [44] CUENCA, E., SALLABERRY, A., YING WANG, F., AND PONCELET, P. Multi-Stream: A Multiresolution Streamgraph Approach to Explore Hierarchical Time Series. *IEEE Transactions on Visualization and Computer Graphics* (2018).
- [45] CUI, W., LIU, S., TAN, L., SHI, C., SONG, Y., GAO, Z., QU, H., AND TONG, X. TextFlow: Towards better understanding of evolving topics in text. *IEEE Trans. Vis. Comput. Graph.* 17, 12 (2011), 2412–2421.
- [46] CUI, W., WU, Y., LIU, S., WEI, F., ZHOU, M. X., AND QU, H. Context preserving dynamic word cloud visualization. In *Visualization Symposium (PacificVis), 2010 IEEE Pacific* (2010), IEEE, pp. 121–128.

- [47] CWSANDBOX. CWSandbox - Automated Online Malware Analysis, Understanding The Sandbox Concept of Malware Identification. <http://cwsandbox.org/>, Accessed 2020.
- [48] DANG, T., NGUYEN, H. N., AND NGUYEN, N. V. VixLSTM: Visual Explainable LSTM for Multivariate Time Series. In *The 12th International Conference on Advances in Information Technology* (2021), pp. 1–5.
- [49] DANG, T., NGUYEN, H. N., AND PHAM, V. WordStream: Interactive Visualization for Topic Evolution. In *EuroVis 2019 - Short Papers* (2019), J. Johansson, F. Sadlo, and G. E. Marai, Eds., The Eurographics Association.
- [50] DANG, T., NGUYEN, N., AND CHEN, Y. Hiperview: real-time monitoring of dynamic behaviors of high-performance computing centers. *The Journal of Supercomputing* 77 (2021), 11807–11826.
- [51] DANG, T., NGUYEN, N. V., LI, J., SILL, A., HASS, J., AND CHEN, Y. Jobviewer: Graph-based visualization for monitoring high-performance computing system. In *2022 IEEE/ACM International Conference on Big Data Computing, Applications and Technologies (BDCAT)* (2022), IEEE, pp. 110–119.
- [52] DANG, T., AND NGUYEN, V. T. ComModeler: Topic Modeling Using Community Detection. In *EuroVis Workshop on Visual Analytics (EuroVA)* (2018), C. Tominski and T. von Landesberger, Eds., The Eurographics Association.
- [53] DANG, T., PHAM, V., NGUYEN, H. N., AND NGUYEN, N. V. AgasedViz: visualizing groundwater availability of Ogallala Aquifer, USA. *Environmental Earth Sciences* 79, 5 (2020), 1–12.
- [54] DANG, T., VAN, H., NGUYEN, H., PHAM, V., AND HEWETT, R. DeepVix: explaining long short-term memory network with high dimensional time series data. In *Proceedings of the 11th International Conference on Advances in Information Technology* (2020), pp. 1–10.
- [55] DANG, T., AND WILKINSON, L. TimeExplorer: Similarity search time series by their signatures. In *Proc. International Symp. on Visual Computing* (2013), pp. 280–289.
- [56] DANG, T. N., ANAND, A., AND WILKINSON, L. TimeSeer: Scagnostics for high-dimensional time series. *IEEE Trans. Vis. Comput. Graph.* 19, 3 (March 2013), 470–483.
- [57] DANG, T. N., PENDAR, N., AND FORBES, A. G. Timearcs: Visualizing fluctuations in dynamic networks. In *Computer Graphics Forum* (2016), vol. 35, Wiley Online Library, pp. 61–69.
- [58] DASU, T., AND JOHNSON, T. *Exploratory data mining and data cleaning*. John Wiley & Sons, 2003.

- [59] DAVIES, J. Word Cloud Generator. <https://www.jasondavies.com/wordcloud/>, 2015. [Online; accessed August 4, 2022].
- [60] DAVIS, F. D. *A technology acceptance model for empirically testing new end-user information systems: Theory and results*. PhD thesis, Massachusetts Institute of Technology, 1985.
- [61] DAVIS, F. D. Perceived usefulness, perceived ease of use, and user acceptance of information technology. *MIS quarterly* (1989), 319–340.
- [62] DEN BROECK, W., CATTUTO, C., BARRAT, A., COLIZZA, V., PAOLOTTI, D., PINTON, J.-F., AND VESPIGNANI, A. Exposing contact patterns. <http://www.sociopatterns.org/2008/06/exposing-contact-patterns/>, 2008.
- [63] DIMARA, E., AND PERIN, C. What is interaction for data visualization? *IEEE transactions on visualization and computer graphics* 26, 1 (2019), 119–129.
- [64] DINABURG, A., ROYAL, P., SHARIF, M., AND LEE, W. Ether: malware analysis via hardware virtualization extensions. In *Proceedings of the 15th ACM conference on Computer and communications security* (2008), pp. 51–62.
- [65] DONAHUE, J., PATURI, A., AND MUKKAMALA, S. Visualization techniques for efficient malware detection. In *2013 IEEE International Conference on Intelligence and Security Informatics* (2013), IEEE, pp. 289–291.
- [66] DÖRK, M., GRUEN, D., WILLIAMSON, C., AND CARPENDALE, S. A Visual backchannel for large-scale events. *IEEE Transactions on Visualization and Computer Graphics* 16, 6 (2010), 1129–1138.
- [67] DOU, W., WANG, X., SKAU, D., RIBARSKY, W., AND ZHOU, M. X. Leadline: Interactive visual analysis of text data through event identification and exploration. In *2012 IEEE Conference on Visual Analytics Science and Technology (VAST)* (2012), pp. 93–102.
- [68] DURANT, K. T., MCCRAY, A. T., AND SAFRAN, C. Modeling the temporal evolution of an online cancer forum. In *Proceedings of the 1st ACM International Health Informatics Symposium* (2010), pp. 356–365.
- [69] D’ORAZIO, F. The future of social media research. <https://www.pulsarplatform.com/resources/the-future-of-social-media-research/>, 2013. [Online; accessed August 5, 2022].
- [70] ELMQVIST, N., AND FEKETE, J.-D. Hierarchical aggregation for information visualization: Overview, techniques, and design guidelines. *IEEE Transactions on Visualization and Computer Graphics* 16, 3 (2010), 439–454.
- [71] FEINBERG, J. Wordle. *Beautiful Visualization: Looking at data through the eyes of experts*. (2010), 37–58.

- [72] FELIX, C., FRANCONERI, S., AND BERTINI, E. Taking word clouds apart: An empirical investigation of the design space for keyword summaries. *IEEE transactions on visualization and computer graphics* 24, 1 (2018), 657–666.
- [73] FENG, T., YANG, J., EPPES, M.-C., YANG, Z., AND MOSER, F. Evis: Visually analyzing environmentally driven events. *IEEE Transactions on Visualization and Computer Graphics* 28, 1 (2021), 912–921.
- [74] FEW, S., AND EDGE, P. Data visualization: past, present, and future. *IBM Cognos Innovation Center* (2007).
- [75] FRIENDLY, M., AND DENIS, D. J. Milestones in the history of thematic cartography, statistical graphics, and data visualization.
- [76] GAMA, J., SEBASTIAO, R., AND RODRIGUES, P. P. Issues in Evaluation of Stream Learning Algorithms. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2009), pp. 329–338.
- [77] GLOOR, P., LAUBACHER, R., ZHAO, Y., AND DYNES, S. Temporal visualization and analysis of social networks. In *NAACSOS Conference, June* (2004), pp. 27–29.
- [78] GLOOR, P. A., AND ZHAO, Y. Analyzing actors and their discussion topics by semantic social network analysis. In *Tenth International Conference on Information Visualisation (IV'06)* (2006), IEEE, pp. 130–135.
- [79] GOTZ, D., AND ZHOU, M. X. Characterizing users’ visual analytic activity for insight provenance. In *2008 IEEE Symposium on Visual Analytics Science and Technology* (2008), IEEE, pp. 123–130.
- [80] GOVE, R., SAXE, J., GOLD, S., LONG, A., AND BERGAMO, G. Seem: a scalable visualization for comparing multiple large sets of attributes for malware analysis. In *Proceedings of the Eleventh Workshop on Visualization for Cyber Security* (2014), pp. 72–79.
- [81] GRÉGIO, A. R., FERNANDES FILHO, D. S., AFONSO, V. M., SANTOS, R. D., JINO, M., AND DE GEUS, P. L. Behavioral analysis of malicious code through network traffic and system call monitoring. In *Evolutionary and Bio-Inspired Computation: Theory and Applications V* (2011), vol. 8059, International Society for Optics and Photonics, p. 805900.
- [82] GRÉGIO, A. R. A., BARUQUE, A. O. C., AFONSO, V. M., FERNANDES FILHO, D. S., DE GEUS, P. L., JINO, M., AND DOS SANTOS, R. D. C. Interactive, visual-aided tools to analyze malware behavior. In *International Conference on Computational Science and Its Applications* (2012), Springer, pp. 302–313.

- [83] GRÉGIO, A. R. A., AND SANTOS, R. D. C. Visualization techniques for malware behavior analysis. In *Sensors, and Command, Control, Communications, and Intelligence (C3I) Technologies for Homeland Security and Homeland Defense X* (2011), E. M. Carapezza, Ed., vol. 8019, International Society for Optics and Photonics, SPIE, pp. 9 – 17.
- [84] GREGORY, M., AND SHNEIDERMAN, B. *Shape Identification in Temporal Data Sets*. Springer London, London, 2012, pp. 305–321.
- [85] GUARNIERI, C. Cuckoo Sandbox: Automated Malware Analysis. <https://cuckoosandbox.org/>, Accessed 2019.
- [86] GUBA, E. G., LINCOLN, Y. S., ET AL. Competing paradigms in qualitative research. *Handbook of qualitative research 2*, 163-194 (1994), 105.
- [87] GUO, Y., GUO, S., JIN, Z., KAUL, S., GOTZ, D., AND CAO, N. Survey on visual analysis of event sequence data. *IEEE Transactions on Visualization and Computer Graphics* 28, 12 (2022), 5091–5112.
- [88] HACKL, J., SCHOLTES, I., PETROVIĆ, L. V., PERRI, V., VERGINER, L., AND GOTE, C. Analysis and visualisation of time series data on networks with pathpy. In *Companion Proceedings of the Web Conference 2021* (New York, NY, USA, 2021), WWW '21, Association for Computing Machinery, p. 530–532.
- [89] HAN, K., KANG, B., AND IM, E. G. Malware analysis using visualized image matrices. *The Scientific World Journal* 2014 (2014).
- [90] HAN, K., LIM, J. H., AND IM, E. G. Malware analysis method using visualization of binary files. In *The 2013 Research in Adaptive and Convergent Systems* (2013), ACM, pp. 317–321.
- [91] HARRIS, R. L. *Information Graphics: A Comprehensive Illustrated Reference*. Oxford University Press, 2000.
- [92] HAVRE, S., HETZLER, B., AND NOWELL, L. ThemeRiver: visualizing theme changes over time. *Proceedings of IEEE Symposium on Information Visualization 2000. INFOVIS 2000 2000* (2000), 115–123.
- [93] HENDERSON, S., AND SEGAL, E. H. Visualizing qualitative data in evaluation research. *New Directions for Evaluation* 2013, 139 (2013), 53–71.
- [94] HISPASEC SISTEMAS. VirusTotal Public API v2.0., Getting started with v2. <https://www.virustotal.com/en/documentation/public-api/>, 2019.
- [95] HONNIBAL, M., MONTANI, I., VAN LANDEGHEM, S., AND BOYD, A. spaCy: Industrial-strength Natural Language Processing in Python, 2020.
- [96] HOQUE, E., SETLUR, V., TORY, M., AND DYKEMAN, I. Applying pragmatics principles for interaction with visual analytics. *IEEE transactions on visualization and computer graphics* 24, 1 (2017), 309–318.

- [97] HULLMAN, J., ADAR, E., AND SHAH, P. Benefitting infovis with visual difficulties. *IEEE Transactions on Visualization and Computer Graphics* 17, 12 (2011), 2213–2222.
- [98] IMPERVA. What is a Backdoor Attack, Shell and Trojan Removal, Imperva. <https://www.imperva.com/learn/application-security/backdoor-shell-attack/>, Accessed 2019.
- [99] INFOSEC. Windows Functions in Malware Analysis Cheat Sheet Part 1. <https://resources.infosecinstitute.com/windows-functions-in-malware-analysis-cheat-sheet-part-1/>, 2015. (Accessed: 2019-06-05).
- [100] INFOSEC. Windows Functions in Malware Analysis Cheat Sheet Part 2. <https://resources.infosecinstitute.com/windows-functions-in-malware-analysis-cheat-sheet-part-2/>, 2015. (Accessed: 2019-06-05).
- [101] ISENBERG, P., HEIMERL, F., KOCH, S., ISENBERG, T., XU, P., STOLPER, C., SEDLMAIR, M., CHEN, J., MÖLLER, T., AND STASKO, J. vispubdata.org: A Metadata Collection about IEEE Visualization (VIS) Publications. *IEEE Transactions on Visualization and Computer Graphics* 23, 9 (Sept. 2017), 2199–2206.
- [102] JENKINS, J., AND CAI, H. Dissecting Android Inter-component Communications via Interactive Visual Explorations. In *2017 IEEE International Conference on Software Maintenance and Evolution (ICSME)* (2017), IEEE, pp. 519–523.
- [103] JENKINS, J., AND CAI, H. ICC-Inspect: Supporting Runtime Inspection of Android Inter-Component Communications. In *2018 IEEE/ACM 5th International Conference on Mobile Software Engineering and Systems (MOBILESoft)* (2018), pp. 80–83.
- [104] JO, J., LEE, B., AND SEO, J. WordlePlus: Expanding Wordle’s Use through Natural Interaction and Animation. *IEEE Computer Graphics and Applications* 35, 6 (2015), 20–28.
- [105] JORDÃO, V., GONÇALVES, D., AND GAMA, S. Eduvis: Visualizing educational information. In *Proceedings of the 8th Nordic Conference on Human-Computer Interaction: Fun, Fast, Foundational* (New York, NY, USA, 2014), NordiCHI ’14, Association for Computing Machinery, p. 1011–1014.
- [106] KANCHERLA, K., DONAHUE, J., AND MUKKAMALA, S. Packer identification using byte plot and markov plot. *Journal of Computer Virology and Hacking Techniques* 12, 2 (2016), 101–111.
- [107] KANCHERLA, K., AND MUKKAMALA, S. Image visualization based malware detection. In *2013 IEEE Symposium on Computational Intelligence in Cyber Security (CICS)* (2013), IEEE, pp. 40–44.

- [108] KANDEL, S., HEER, J., PLAISANT, C., KENNEDY, J., VAN HAM, F., RICHEL, N. H., WEAVER, C., LEE, B., BRODBECK, D., AND BUONO, P. Research directions in data wrangling: Visualizations and transformations for usable and credible data. *Information Visualization* 10, 4 (2011), 271–288.
- [109] KANG, H., GETOOR, L., AND SINGH, L. Visual analysis of dynamic group membership in temporal social networks. *ACM SIGKDD Explorations Newsletter* 9, 2 (2007), 13–21.
- [110] KASPERSKY. What is a Trojan horse and what damage can it do? - Kaspersky. <https://usa.kaspersky.com/resource-center/threats/trojans>, Online; accessed July 31, 2023.
- [111] KEIM, D., ANDRIENKO, G., FEKETE, J.-D., GÖRG, C., KOHLHAMMER, J., AND MELANÇON, G. *Visual Analytics: Definition, Process, and Challenges*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008, pp. 154–175.
- [112] KELLY, S. Modest natural-language processing. <https://compromise.cool/>, 2019. [Online; accessed August 5, 2022].
- [113] KNIGGE, L., AND COPE, M. Grounded Visualization: Integrating the Analysis of Qualitative and Quantitative Data through Grounded Theory and Visualization. *Environment and Planning A: Economy and Space* 38, 11 (2006), 2021–2037.
- [114] KOH, K., LEE, B., KIM, B., AND SEO, J. Maniwordle: Providing flexible control over wordle. *IEEE Transactions on Visualization and Computer Graphics* 16, 6 (2010), 1190–1197.
- [115] KOSARA, R. Data Visualization Fundamentals and Best Practices, Lesson 1: Introduction. <https://observablehq.com/@observablehq/datavis-course-lesson-1-introduction>, Mar 2023.
- [116] KRSTAJIC, M., BERTINI, E., AND KEIM, D. Cloudlines: Compact display of event episodes in multiple time-series. *IEEE Transactions on Visualization and Computer Graphics* 17, 12 (2011), 2432–2439.
- [117] KUCHER, K., SULTANUM, N., DAZA, A., SIMAKI, V., SKEPPSTEDT, M., PLANK, B., FEKETE, J.-D., AND MAHYAR, N. An interdisciplinary perspective on evaluation and experimental design for visual text analytics: Position paper. In *2022 IEEE Evaluation and Beyond - Methodological Approaches for Visualization (BELIV)* (2022), pp. 28–37.
- [118] KUMAR, S., MADHAVAN, L., NAGAPPAN, M., AND SIKDAR, B. Malware in pirated software: Case study of malware encounters in personal computers. In *2016 11th International Conference on Availability, Reliability and Security (ARES)* (2016), IEEE, pp. 423–427.
- [119] LAMBIOTTE, R., ROSVALL, M., AND SCHOLTES, I. From networks to optimal higher-order models of complex systems. *Nature physics* 15, 4 (2019), 313–320.

- [120] LAWRENCE LIVERMORE NATIONAL LABORATORY. Rose compiler: Program analysis and transformation. <http://rosecompiler.org/>, Accessed 2010.
- [121] LE, D. D., PHAM, V., NGUYEN, H. N., AND DANG, T. Visualization and explainable machine learning for efficient manufacturing and system operations. *Smart and Sustainable Manufacturing Systems* 3, 2 (2019), 127–147.
- [122] LEDOUX, C., AND LAKHOTIA, A. Malware and machine learning. In *Intelligent Methods for Cyber Warfare*. Springer, 2015, pp. 1–42.
- [123] LENGUEL, T. K., MARESCA, S., PAYNE, B. D., WEBSTER, G. D., VOGL, S., AND KIAYIAS, A. Scalability, fidelity and stealth in the drakvuf dynamic malware analysis system. In *The 30th Annual Computer Security Applications Conference* (2014), pp. 386–395.
- [124] LIU, M., LI, Q., LIANG, L., LI, J., WANG, K., LI, J., LV, M., CHEN, N., SONG, H., LEE, J., ET AL. Real-time visualization of clustering and intracellular transport of gold nanoparticles by correlative imaging. *Nature communications* 8, 1 (2017), 15646.
- [125] LIU, S., ZHOU, M. X., PAN, S., SONG, Y., QIAN, W., CAI, W., AND LIAN, X. TIARA: interactive, topic-based visual text summarization and analysis. *ACM TIST* 3, 2 (2012), 25:1–25:28.
- [126] LOHMANN, S., ZIEGLER, J., AND TETZLAFF, L. Comparison of tag cloud layouts: Task-related performance and visual exploration. In *Proceedings of the 12th IFIP TC 13 International Conference on Human-Computer Interaction: Part I* (Berlin, Heidelberg, 2009), INTERACT '09, Springer-Verlag, pp. 392–404.
- [127] LONG, A., SAXE, J., AND GOVE, R. Detecting malware samples with similar image sets. In *Proceedings of the Eleventh Workshop on Visualization for Cyber Security* (2014), pp. 88–95.
- [128] LYI, S., WANG, Q., LEKSCHAS, F., AND GEHLENBORG, N. Gosling: A grammar-based toolkit for scalable and interactive genomics data visualization. *IEEE Transactions on Visualization and Computer Graphics* 28, 1 (2021), 140–150.
- [129] MAGALLANES, J., VAN GEMEREN, L., WOOD, S., AND VILLA-URIOL, M.-C. Analyzing time attributes in temporal event sequences. In *2019 IEEE Visualization Conference (VIS)* (2019), IEEE, pp. 1–5.
- [130] MALWAREBYTES LABS. Kaseya hijacked, thousands attacked by revil, fix delayed again. <https://blog.malwarebytes.com/cybercrime/2021/07/shutdown-kaseya-vsa-servers-now-amidst-cascading-revil-attack-against-msps-clients/>, 2021.

- [131] MAURI, M., ELLI, T., CAVIGLIA, G., UBOLDI, G., AND AZZI, M. Rawgraphs: a visualisation platform to create open outputs. In *Proceedings of the 12th biannual conference on Italian SIGCHI chapter* (2017), pp. 1–5.
- [132] MICROSOFT SECURITY INTELLIGENCE. Trojan:Win32/MultiInjector.C. <https://www.microsoft.com/en-us/wdsi/threats/malware-encyclopedia-description?Name=Trojan:Win32/MultiInjector.C>. Accessed: 2022-05-24.
- [133] MCGRAW, T. Glitch style visualization of disrupted neuronal connectivity in parkinson’s disease. In *2017 IEEE VIS Arts Program (VISAP)* (2017), IEEE, pp. 1–8.
- [134] MCLACHLAN, P., MUNZNER, T., KOUTSOFIOS, E., AND NORTH, S. Liverac: interactive visual exploration of system management time-series data. In *Proceedings of the SIGCHI conference on human factors in computing systems* (2008), pp. 1483–1492.
- [135] MICROSOFT. Cyberthreats, viruses, and malware - Microsoft Security Intelligence. <https://www.microsoft.com/en-us/wdsi/threats>. Accessed: 2019-12-24.
- [136] MICROSOFT. How Microsoft identifies malware and potentially unwanted applications. <https://docs.microsoft.com/en-us/windows/security/threat-protection/intelligence/criteria>, 2020. Accessed: 2020-11-29.
- [137] MICROSOFT. Malware Names. <https://docs.microsoft.com/en-us/windows/security/threat-protection/intelligence/malware-naming>, 2020. Accessed: 2020-11-29.
- [138] MICROSOFT. Process Monitor - Windows Sysinternals - Microsoft Docs, 2020. Accessed: 2020-11-29. <https://docs.microsoft.com/en-us/sysinternals/downloads/procmon>.
- [139] MICROSOFT SECURITY INTELLIGENCE. Behavior:Win32/Bladabindi.gen. <https://www.microsoft.com/en-us/wdsi/threats/malware-encyclopedia-description?Name=Behavior:Win32/Bladabindi.gen&threatId=-2147281575>. Accessed: 2019-12-24.
- [140] MICROSOFT SECURITY INTELLIGENCE. Behavior:Win32/MultiInjector. <https://wdsi-filesubmission.trafficmanager.net/en-us/wdsi/threats/malware-encyclopedia-description?Name=Behavior:Win32/MultiInjector&threatId=-2147325646>. Accessed: 2019-12-24.
- [141] MICROSOFT SECURITY INTELLIGENCE. Behavior:Win32/Teerac.B. <https://wdsi-filesubmission.trafficmanager.net/en-us/wdsi/threats/malware-encyclopedia-description?Name=Behavior:Win32/Teerac.B&threatId=-2147277970>. Accessed: 2019-12-24.

- [142] MICROSOFT SECURITY INTELLIGENCE. Behavior:Win32/Vawtrak.A. <https://wdsi-filesubmission.trafficmanager.net/en-us/wdsi/threats/malware-encyclopedia-description?Name=Behavior:Win32/Vawtrak.A&threatId=-2147280787>. Accessed: 2019-12-24.
- [143] MICROSOFT SECURITY INTELLIGENCE. HackTool:Win32/Mailpassview. <https://www.microsoft.com/en-us/wdsi/threats/malware-encyclopedia-description?Name=HackTool:Win32/Mailpassview&threatId=-2147395884>. Accessed: 2019-12-24.
- [144] MILES, C., LAKHOTIA, A., LEDOUX, C., NEWSOM, A., AND NOTANI, V. Virusbattle: State-of-the-art malware analysis for better cyber threat intelligence. In *2014 7th International Symposium on Resilient Control Systems (ISRCs)* (2014), IEEE, pp. 1–6.
- [145] MITRE CORPORATION. Mitre att&ck framework. <https://attack.mitre.org/>, Accessed 2019.
- [146] MONGE, P. R., AND CONTRACTOR, N. S. *Theories of communication networks*. Oxford University Press, USA, 2003.
- [147] MONTANI, I. An open-source named entity visualiser for the modern web, 2017. <https://explosion.ai/blog/displacy-ent-named-entity-visualizer> [Accessed date: April 10, 2019].
- [148] MOODY, J., MCFARLAND, D., AND BENDER-DEMOLL, S. Dynamic network visualization. *American journal of sociology* 110, 4 (2005), 1206–1241.
- [149] MORRIS, M., KURTH, A. E., HAMILTON, D. T., MOODY, J., AND WAKEFIELD, S. Concurrent partnerships and hiv prevalence disparities by race: linking science and public health practice. *American journal of public health* 99, 6 (2009), 1023–1031.
- [150] MUNZNER, T. A nested model for visualization design and validation. *IEEE Transactions on Visualization and Computer Graphics* 15, 6 (2009), 921–928.
- [151] MUNZNER, T. *Visualization analysis and design*. CRC press, 2014.
- [152] MUTTON, P. Inferring and visualizing social networks on internet relay chat. In *Proceedings. Eighth International Conference on Information Visualisation, 2004. IV 2004.* (2004), IEEE, pp. 35–43.
- [153] NARECHANIA, A., SRINIVASAN, A., AND STASKO, J. Nl4dv: A toolkit for generating analytic specifications for data visualization from natural language queries. *IEEE Transactions on Visualization and Computer Graphics* 27, 2 (2020), 369–379.
- [154] NGUYEN, H. N. WordStream Library. <https://github.com/huyen-nguyen/wordstream-library>, 2019.

- [155] NGUYEN, H. N., ABRI, F., PHAM, V., CHATTERJEE, M., NAMIN, A. S., AND DANG, T. Malview: Interactive visual analytics for comprehending malware behavior. *IEEE Access* 10 (2022), 99909–99930.
- [156] NGUYEN, H. N., ABRI, F., PHAM, V., CHATTERJEE, M., NAMIN, A. S., AND DANG, T. Malview: Interactive visual analytics for comprehending malware behavior. *IEEE Access* (2022).
- [157] NGUYEN, H. N., AND DANG, T. EQSA: Earthquake Situational Analytics from Social Media. In *2019 IEEE Conference on Visual Analytics Science and Technology (VAST)* (2019), pp. 142–143.
- [158] NGUYEN, H. N., AND DANG, T. TTU-Nguyen-MC3, IEEE Visual Analytics Science and Technology (VAST) Challenge 2019 Mini-Challenge 3, 2019. <https://idatavisualizationlab.github.io/VAST2019mc3/TTU-Nguyen-MC3/index.htm> [Accessed date: July 3, 2023].
- [159] NGUYEN, H. N., DANG, T., AND BOWE, K. A. WordStream Maker: A Lightweight End-to-end Visualization Platform for Qualitative Time-series Data. *NLVIZ: Exploring Research Opportunities for Natural Language, Text, and Data Visualization, IEEE VIS 2022* (2022).
- [160] NGUYEN, H. N., NGUYEN, V. T., AND DANG, T. Interface design for hci classroom: From learners’ perspective. In *International Symposium on Visual Computing* (2020), Springer, pp. 545–557.
- [161] NGUYEN, H. N., TRUJILLO, C. M., WEE, K., AND BOWE, K. A. Interactive Qualitative Data Visualization for Educational Assessment. In *The 12th International Conference on Advances in Information Technology* (New York, NY, USA, 2021), IAIT2021, ACM.
- [162] NGUYEN, N. V., NGUYEN, H. N., HASS, J., AND DANG, T. JobNet: 2D and 3D Visualization for Temporal and Structural Association in High-Performance Computing System. In *International Symposium on Visual Computing* (2021), Springer, pp. 210–221.
- [163] NGUYEN, P. H., HENKIN, R., CHEN, S., ANDRIENKO, N., ANDRIENKO, G., THONNARD, O., AND TURKAY, C. Vasabi: Hierarchical user profiles for interactive visual user behaviour analytics. *IEEE transactions on visualization and computer graphics* 26, 1 (2019), 77–86.
- [164] NGUYEN, V. T., NAMIN, A. S., AND DANG, T. Malviz: An interactive visualization tool for tracing malware. In *Proceedings of the 27th ACM SIGSOFT International Symposium on Software Testing and Analysis* (New York, NY, USA, 2018), ISSTA 2018, ACM, pp. 376–379.
- [165] NIGHTINGALE, F. *Notes on Matters Affecting the Health, Efficiency and Hospital Administration of the British Army*. Harrison and Sons, 1858.

- [166] NIU, Z., CHENG, D., ZHANG, L., AND ZHANG, J. Visual analytics for networked-guarantee loans risk management. In *2018 IEEE Pacific Visualization Symposium (PacificVis)* (2018), pp. 160–169.
- [167] O’GORMAN, G., AND McDONALD, G. Ransomware: A growing menace. Symantec: Security Response.
- [168] OPSWAT, INC. Metadefender cloud. <https://metadefender.opswat.com/>, 2020.
- [169] O’SHAUGHNESSY, S. Image-based malware classification: A space filling curve approach. In *2019 IEEE Symposium on Visualization for Cyber Security (VizSec)* (2019), IEEE, pp. 1–10.
- [170] OTTLEY, A., KASZOWSKA, A., CROUSER, R. J., AND PECK, E. M. The curious case of combining text and visualization. In *EuroVis (Short Papers)* (2019), pp. 121–125.
- [171] PANAS, T. Signature visualization of software binaries. In *Proceedings of the 4th ACM Symposium on Software Visualization* (New York, NY, USA, 2008), ACM, pp. 185–188.
- [172] PATURI, A., CHERUKURI, M., DONAHUE, J., AND MUKKAMALA, S. Mobile malware visual analytics and similarities of attack toolkits (malware gene analysis). In *2013 International Conference on Collaboration Technologies and Systems (CTS)* (2013), IEEE, pp. 149–154.
- [173] PERER, A., GUY, I., UZIEL, E., RONEN, I., AND JACOVI, M. The longitudinal use of sandvis: Visual social network analytics in the enterprise. *IEEE Transactions on Visualization and Computer Graphics* 19, 7 (2013), 1095–1108.
- [174] PERER, A., AND SHNEIDERMAN, B. Balancing systematic and flexible exploration of social networks. *IEEE transactions on visualization and computer graphics* 12, 5 (2006), 693–700.
- [175] PERLMAN, G. Perceived Usefulness and Ease of Use, Web-Based Questionnaires. <https://garyperلمان.com/quest/quest.cgi?form=PUEU>.
- [176] PHAM, D. V., SYED, A., AND HALGAMUGE, M. N. Universal serial bus based software attacks and protection solutions. *digital investigation* 7, 3-4 (2011), 172–184.
- [177] PHAM, V., AND DANG, T. Cvexplorer: Multidimensional visualization for common vulnerabilities and exposures. In *2018 IEEE International Conference on Big Data (Big Data)* (Dec 2018), pp. 1296–1301.
- [178] POKORNY, J. J., NORMAN, A., ZANESCO, A. P., BAUER-WU, S., SAHDRA, B. K., AND SARON, C. D. Network analysis for the visualization and analysis of qualitative data. *Psychological methods* 23, 1 (2018), 169.

- [179] POTMA, E. O., DE BOEIJ, W. P., VAN HAASTERT, P. J., AND WIERSMA, D. A. Real-time visualization of intracellular hydrodynamics in single living cells. *Proceedings of the National Academy of Sciences* 98, 4 (2001), 1577–1582.
- [180] QAISER, S., AND ALI, R. Text mining: use of tf-idf to examine the relevance of words to documents. *International Journal of Computer Applications* 181, 1 (2018), 25–29.
- [181] QUICK HEAL THREAT RESEARCH & RESPONSE LABS. ‘Golroted’ malware uses web browser weakness to steal sensitive information. http://dlupdate.quickheal.com/documents/others/quick_heal_golroted_malware_threat_report_june_2015.pdf. Accessed: 2019-12-24.
- [182] QUIST, D. A., AND LIEBROCK, L. M. Visualizing compiled executables for malware analysis. In *2009 6th International Workshop on Visualization for Cyber Security* (2009), pp. 27–32.
- [183] QUIST, D. A., AND LIEBROCK, L. M. Reversing compiled executables for malware analysis via visualization. *Information Visualization* 10, 2 (2011), 117–126.
- [184] REDMON, J., DIVVALA, S., GIRSHICK, R., AND FARHADI, A. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2016).
- [185] RUSSINOVICH, M. E., AND MARGOSIS, A. *Windows Sysinternals Administrator’s Reference*, 1st ed. Microsoft Press, Redmond, WA, USA, 2011.
- [186] SACHA, D., SENARATNE, H., KWON, B. C., ELLIS, G., AND KEIM, D. A. The role of uncertainty, awareness, and trust in visual analytics. *IEEE Transactions on Visualization and Computer Graphics* 22, 1 (2016), 240–249.
- [187] SAKET, B., KIM, H., BROWN, E. T., AND ENDERT, A. Visualization by demonstration: An interaction paradigm for visual data exploration. *IEEE transactions on visualization and computer graphics* 23, 1 (2016), 331–340.
- [188] SATYANARAYAN, A., MORITZ, D., WONGSUPHASAWAT, K., AND HEER, J. Vega-lite: A grammar of interactive graphics. *IEEE transactions on visualization and computer graphics* 23, 1 (2016), 341–350.
- [189] SAXE, J., MENTIS, D., AND GREAMO, C. Visualization of shared system call sequence relationships in large malware corpora. In *Proceedings of the Ninth International Symposium on Visualization for Cyber Security* (NY, USA, 2012), ACM, pp. 33–40.
- [190] SCHÜTZE, H., MANNING, C. D., AND RAGHAVAN, P. *Introduction to information retrieval*, vol. 39. Cambridge University Press, 2008.

- [191] SECURITY, P. Free automated malware analysis service - hybrid analysis. <https://www.hybrid-analysis.com/>, Accessed 2019.
- [192] SETLUR, V., BATTERSBY, S. E., TORY, M., GOSSWEILER, R., AND CHANG, A. X. Eviza: A natural language interface for visual analysis. In *Proceedings of the 29th annual symposium on user interface software and technology* (2016), pp. 365–377.
- [193] SHAFFER, D. W., HATFIELD, D., SVAROVSKY, G. N., NASH, P., NULTY, A., BAGLEY, E., FRANK, K., RUPP, A. A., AND MISLEVY, R. Epistemic network analysis: A prototype for 21st-century assessment of learning. *International Journal of Learning and Media* 1, 2 (2009).
- [194] SHAID, S. Z. M., AND MAAROF, M. A. Malware behavior image for malware variant identification. In *2014 International Symposium on Biometrics and Security Technologies (ISBAST)* (2014), IEEE, pp. 238–243.
- [195] SHAID, S. Z. M., AND MAAROF, M. A. Malware behaviour visualization. *Jurnal Teknologi* 70, 5 (2014).
- [196] SHAMMA, D. A., KENNEDY, L., AND CHURCHILL, E. F. Peaks and persistence: modeling the shape of microblog conversations. In *Proceedings of the ACM 2011 conference on Computer supported cooperative work* (2011), pp. 355–358.
- [197] SHNEIDERMAN, B. The eyes have it: A task by data type taxonomy for information visualizations. In *Proceedings 1996 IEEE symposium on visual languages* (1996), IEEE, pp. 336–343.
- [198] SPARK, J. Time chart of world history: A histomap of peoples and nations for 4,000 years, 1931. <http://www.worldhistorycharts.com/the-histomap-by-john-sparks/> [Accessed date: Feb 20, 2019].
- [199] SPRING, J. M., METCALF, L. B., AND STONER, E. Correlating domain registrations and dns first activity in general and for malware. In *Securing and Trusting Internet Names: SATIN 2011* (2011), National Physical Laboratory.
- [200] SRINIVASAN, A., AND STASKO, J. How to ask what to say?: Strategies for evaluating natural language interfaces for data visualization. *IEEE Computer Graphics and Applications* 40, 4 (2020), 96–103.
- [201] STOLTE, C., TANG, D., AND HANRAHAN, P. Polaris: A system for query, analysis, and visualization of multidimensional relational databases. *IEEE Transactions on Visualization and Computer Graphics* 8, 1 (2002), 52–65.
- [202] STROBELT, H., SPICKER, M., STOFFEL, A., KEIM, D., AND DEUSSEN, O. Rolled-out Wordles: A Heuristic Method for Overlap Removal of 2D Data Representatives. *Computer Graphics Forum* 31, 3pt3 (2012), 1135–1144.

- [203] SUN, G., WU, Y., LIU, S., PENG, T., ZHU, J. J. H., AND LIANG, R. Evoriver: Visual analysis of topic coopetition on social media. *IEEE Transactions on Visualization and Computer Graphics* 20, 12 (Dec 2014), 1753–1762.
- [204] TABASSUM, S., PEREIRA, F. S., AND GAMA, J. Knowledge discovery from temporal social networks. *Intelligent Informatics* (2018), 10.
- [205] THOMPSON, K., ASHE, D., CARVALHO, L., GOODYEAR, P., KELLY, N., AND PARISIO, M. Processing and visualizing data in complex learning environments. *American Behavioral Scientist* 57, 10 (2013), 1401–1420.
- [206] TRIER, M. Research note—towards dynamic visualization for understanding evolution of digital communication networks. *Information Systems Research* 19, 3 (2008), 335–350.
- [207] TRINIUS, P., HOLZ, T., GÖBEL, J., AND FREILING, F. C. Visual analysis of malware behavior using treemaps and thread graphs. In *2009 6th International Workshop on Visualization for Cyber Security* (2009), IEEE, pp. 33–38.
- [208] TUFTE, E. R. *The visual display of quantitative information*. Graphics Press, 1983.
- [209] TUKEY, J. W., ET AL. *Exploratory data analysis*, vol. 2. Reading, MA, 1977.
- [210] VAN, H., NGUYEN, H. N., HEWETT, R., AND DANG, T. Hackernets: Visualizing media conversations on internet of things, big data, and cybersecurity. In *2019 IEEE International Conference on Big Data (Big Data)* (2019), IEEE, pp. 3293–3302.
- [211] VAN WIJK, J. J. The value of visualization. In *VIS 05. IEEE Visualization, 2005*. (2005), IEEE, pp. 79–86.
- [212] VERDINELLI, S., AND SCAGNOLI, N. I. Data display in qualitative research. *International Journal of Qualitative Methods* 12, 1 (2013), 359–381.
- [213] VIEGAS, F. B., WATTENBERG, M., AND FEINBERG, J. Participatory visualization with wordle. *IEEE transactions on visualization and computer graphics* 15, 6 (2009), 1137–1144.
- [214] WAGNER, M., AIGNER, W., RIND, A., DORNHACKL, H., KADLETZ, K., LUH, R., AND TAVOLATO, P. Problem characterization and abstraction for visual analytics in behavior-based malware pattern analysis. In *Proceedings of the Eleventh Workshop on Visualization for Cyber Security* (2014), pp. 9–16.
- [215] WAGNER, M., FISCHER, F., LUH, R., HABERSON, A., RIND, A., KEIM, D. A., AIGNER, W., BORGO, R., GANOVELLI, F., AND VIOLA, I. A survey of visualization systems for malware analysis. In *EG conference on visualization (EuroVis)-STARs* (2015), The EGA, pp. 105–125.

- [216] WAGNER, M., RIND, A., THÜR, N., AND AIGNER, W. A knowledge-assisted visual malware analysis system: Design, validation, and reflection of kamas. *Computers & Security* 67 (2017), 1–15.
- [217] WALDNER, M., SCHRAMMEL, J., KLEIN, M., KRISTJÁNSDÓTTIR, K., UNGER, D., AND TSCHELIGI, M. Facetclouds: Exploring tag clouds for multi-dimensional data. In *Proceedings of Graphics Interface 2013* (Toronto, Ont., Canada, Canada, 2013), GI '13, Canadian Information Processing Society, pp. 17–24.
- [218] WANG, Y., CHU, X., BAO, C., ZHU, L., DEUSSEN, O., CHEN, B., AND SEDLMAIR, M. EdWordle: Consistency-Preserving Word Cloud Editing. *IEEE Transactions on Visualization and Computer Graphics* 24, 1 (2018), 647–656.
- [219] WANNER, F., STOFFEL, A., JÄCKLE, D., KWON, B. C., WEILER, A., AND KEIM, D. A. State-of-the-Art Report of Visual Analysis for Event Detection in Text Data Streams. In *EuroVis - STARs* (2014), R. Borgo, R. Maciejewski, and I. Viola, Eds., The Eurographics Association.
- [220] WASSERMAN, S., AND FAUST, K. *Social network analysis: Methods and applications*. Cambridge university press, 1994.
- [221] WATTENBERG, M. Baby names, visualization, and social data analysis. In *Proc. IEEE Symp. on Information Visualization* (2005), pp. 1–7.
- [222] WICKHAM, H. ggplot2: Elegant graphics for data analysis. *Media* 35, 211 (2009), 10–1007.
- [223] WICKHAM, H. A layered grammar of graphics. *Journal of Computational and Graphical Statistics* 19, 1 (2010), 3–28.
- [224] WILKINSON, L. *The Grammar of Graphics*. Springer, 2005.
- [225] WILKINSON, L., ANAND, A., AND GROSSMAN, R. High-dimensional visual analytics: Interactive exploration guided by pairwise views of point distributions. *IEEE Transactions on Visualization and Computer Graphics* 12, 6 (2006), 1363–1372.
- [226] XU, Z., ZHANG, J., GU, G., AND LIN, Z. Goldeneye: Efficiently and effectively unveiling malware’s targeted environment. In *International Workshop on Recent Advances in Intrusion Detection* (2014), Springer, pp. 22–45.
- [227] YE, Q., ZHU, T., HU, D., WU, B., DU, N., AND WANG, B. Cell phone mini challenge award: Social network accuracy—exploring temporal communication in mobile call graphs. In *2008 IEEE Symposium on Visual Analytics Science and Technology* (2008), IEEE, pp. 207–208.

- [228] YI, J. S., ELMQVIST, N., AND LEE, S. Timematrix: Analyzing temporal social networks using interactive matrix-based visualizations. *Intl. Journal of Human-Computer Interaction* 26, 11-12 (2010), 1031–1051.
- [229] YOO, I. Visualizing windows executable viruses using self-organizing maps. In *Proceedings of the 2004 ACM Workshop on Visualization and Data Mining for Computer Security* (New York, NY, USA, 2004), ACM, pp. 82–89.
- [230] ZAMORA, W. The state of ransomware among smbs. <https://blog.malwarebytes.com>, 2017.
- [231] ZHANG, S., WANG, Y., LI, H., AND ZHANG, W. Who will support my project? interactive search of potential crowdfunding investors through insearch. In *CHI Conference on Human Factors in Computing Systems Extended Abstracts* (2022), pp. 1–6.
- [232] ZHUO, W., AND NADJIN, Y. Malwarevis: Entity-based visualization of malware network traces. In *Proceedings of the Ninth International Symposium on Visualization for Cyber Security* (New York, NY, USA, 2012), ACM, pp. 41–47.