Thesis Project Portfolio

Development of Dashboard for monitoring the Health of Verizon's Network

(Technical Report)

Analysis of the Tay Chatbot Case through Actor Network Theory

(STS Research Paper)

An Undergraduate Thesis

Presented to the Faculty of the School of Engineering and Applied Science

University of Virginia • Charlottesville, Virginia

In Fulfillment of the Requirements for the Degree

Bachelor of Science, School of Engineering

Shiva Manandhar

Spring, 2023

Department of Computer Science

Table of Contents

Sociotechnical Synthesis

Development of Dashboard for monitoring the Health of Verizon’s Network

Analysis of the Tay Chatbot Case through Actor Network Theory

Prospectus

Socio Technical Synthesis: Redesign of UVA's CS curriculum and Tay Chatbot

My Technical Report and STS Research Paper are connected through the notion that there are overlooked features within technology that can have a major influence on its social aspects. Both projects utilize the ANT framework to examine overlooked features within software development to emphasize the overarching problems within each case. Software development practices are prominent within both my Technical Report and STS Research Paper; both go through the practices that were not emphasized within software development and how they can have social and technical consequences. However, the Technical Report goes over proposed changes within a social construct, UVA's Computer Science Department, that overlook software development practices for groups of individuals. The STS Research Paper delves into the software practices within the Tay Chatbot case that were neglected and caused the Chatbot to become rogue. Even though both focus on different aspects of software engineering, both of the projects are stringed through the ideology of the importance of having proper software development practices.

My Technical Report discussed the proposed reconstruction of UVA's Computer Science curriculum to include new changes based on difficulties I had while working on my Network Dashboard during my internship in Verizon. While at Verizon, I was not familiar with the software development tools that were used to collect data, such as Splunk, and software development practices to ensure the software I am developing is meeting key needs of Verizon engineers, such as asking for feedback on user testing. Due to my lack of understanding of data collection tools and software development practices, I advocated the needs of a redesign within the Computer Science department to ensure that students are more prepared within the workforce when developing software. Specifically, I used ANT to show the need for classes to focus on user experience and software tools to have a better comprehension of software development in the workforce..

In my Technical Report, I analyzed the factors that were overlooked within the Tay Chatbot case to illuminate other areas of software development within Artificial Intelligence that needed to be addressed, such as the developer's testing practices and the algorithm within Tay. Tay was a Chatbot that was developed by Microsoft to facilitate conversations with users on Twitter. The Chatbot, however, developed hostile results due to malicious users taking advantage of Tay's algorithm and teaching it inappropriate language. Many individuals believe that Tay went rogue due to the aggressors on Twitter. But through ANT, I demonstrated that other factors must be taken into consideration, specifically the algorithms used within Tay's software and software practices that were not focused on. Demonstrating there were other actors involved in the downfall of Tay can clarify why Tay did not function properly and lead the public and engineers to focus on areas that are liable to causing Chatbots to become malicious.

Working on both of these projects has improved my ability to understand both software development and the need for software practices. The Technical Report helped me zoom in on overarching topics to learn more about critical details that may need to be readjusted, which lead to me using the Tay Chatbot case to fully understand as to why the Chatbot went rogue. These projects have made me more aware of how technology will interact with consumers, and thus made me aware of making the software and code I write more user friendly. Overall, the projects I have done in my 4th year at UVA helped emphasize the needs of proper software practices to ensure that the software that is being continuously integrated and maintained meets the needs of the consumer while also ensuring that it is functioning properly.

Development of Dashboard for monitoring the Health of Verizon's Network

A Technical Report submitted to the Department of Computer Science

Presented to the Faculty of the School of Engineering and Applied Science

University of Virginia • Charlottesville, Virginia

In Partial Fulfillment of the Requirements for the Degree

Bachelor of Science, School of Engineering

Shiva Manandhar

Spring, 2023

Technical Project Team Members

Shiva Manandhar

On my honor as a University Student, I have neither given nor received unauthorized aid on this assignment as defined by the Honor Guidelines for Thesis-Related Assignments

Kevin Sullivan , Department of Computer Science

Development of Dashboard for monitoring the Health of Verizon's Network

CS4991 Capstone Report, 2022

Shiva Manandhar
Computer Science
The University of Virginia
School of Engineering and Applied Science
Charlottesville, Virginia USA
sm4phs@virginia.edu

ABSTRACT

[Abstract (synopsis of full report in 6-8 sentences; as indicated in general outline below from CS4991 Writing Guide)]

Verizon, a large telephone network based in the United States, needed to create new dashboards to keep track of all the Speed Test routers health to ensure that customers maintain reliable video internet. Customers heavily rely on Verizon's video router for their video services from Fios, and ensuring that the Speed Test routers are active allows Verizon to maintain proper speed and bandwidth for its customers. In my role as an Network Engineer intern, I had to learn how to create a dashboard using Splunk and Python APIs. The dashboard served as a visual implementation notifying Network Engineers when the routers are down and need reconfiguration. Creating this dashboard allowed Verizon to determine which dashboards need reconfiguration, reducing the amount of time needed to find specific routers. Future improvements to the dashboard include developing a map of all router locations and notifying where routers were down through a marker.

1. INTRODUCTION

[Introduction (information providing context/ background for readers)]

Verizon, one of largest Internet Service Providers in the nation, has many routers across the country. These routers often need to be monitored to ensure optimal performance, and issues , such as low bandwidth or loss of packets, need to be reconfigured through an application.

The main application Verizon utilizes to reconfigure routers is called V-repair. V-repair allows network engineers to reconfigure routers, but there are over 1000 routers per each state on average that are being monitored by Verizon, making it difficult to scan between all the routers and check their health individually. Dashboards notify Verizon which routers are currently down, reducing the amount of time necessary to find and repair the routers. Without

Dashboards, Verizon Network Engineers would spend much time manually scanning all the routers, which is an arduous and time extensive task.

To be able to help as many customers as possible and ensure they are having a reliable network, data logs of routers that are stored on databases, such as Splunk, allow Verizon to create dashboard, frontend applications to mitigate the amount of time to find routers that are performing suboptimally.

2. RELATED WORKS

[Related Work (brief literature review of similar apps or concepts)]

Though a good deal of research has been done towards dashboards and their capabilities to transfer data, far less research and development exists for managing network routers and deploying dashboards.

According to Kitchin, et al (2015). Dashboards have been created to monitor the activity of cities by measuring their economic performances and analyzing how well they compare each other (Lauriault, 2010). However, these dashboards are not focused on determining the performance of network routers.

Splunk has been used for analyzing big data sets. Chen and Chien showed that Splunk maintains router health and collects data to maintain Greenhouse Gas Houses' plants and the environment within (Chen, 2017). This demonstrates that Splunk is a very key resource in maintaining infrastructure and identifying trends that can communicate the course of actions needed to maintain optimal performance.

3. PROJECT DESIGN [or PROPOSAL DESIGN or other title as appropriate]

[Process design info (problems proposed app would address and how it would work; divide into numbered and titled subsections 3.1, 3.2, 3.3, etc., if needed for clarity)]

3.1 Review of System Architecture

The dashboard I created at Verizon needed to be able to notify in the frontend which routers need to be reconfigured by Vrepair through the collected logs on Splunk's interface. To ensure this, the dashboard must notify the Network Engineer which router is currently down through a router ID and its current state, allowing the Network Engineer to query for the particular router to be reconfigured. Verizon routers can emit text-parse information, called JSON, which can be stored on a database. The database that Verizon utilizes is called Splunk, and through providing Splunk queries from Python APIs, it is able to query and find information necessary to scan and determine trends from (*Splunk,* n.d). With Splunk, Verizon has the potential to reduce the amount of time spent finding individual routers and, instead, find routers that are not performing well. Thus, through utilizing Splunk, Python APIS, and Splunk UI Language, I was tasked with developing this dashboard through Splunk to ensure that the health of Verizon's routers are adequate with the needs of its customers.

3.2 Requirements

This section goes over the requirements needed for developing the dashboard and which limitations must be addressed while choosing components to be in the dashboard.

3.2.1 Client Needs

The Network Engineers within Verizon, specifically the Network Engineers in Verizon's Network Operation Center in Ashburn, Virginia, are the main clients of this dashboard. In order for them to know which routers need reconfiguration, the dashboard must notify them which routers are down based off of the Cisco security level index displayed on their logs. Through querying syntax from Python to Splunk's Interface, the dashboard is able to display in a list which routers have a specific range of Cisco index level that signifies the router needs reconfiguration. Additionally, the Dashboard should notify the Network Engineer which state has the most routers needing reconfiguration in case there is a severe weather phenomenon, such as a hurricane, within a particular region.

3.2.2 System Limitations

The main limitation of Splunk and the dashboard created through Python APIs and the Splunk Interface is that if Splunk goes down the Network Engineers will not be able to locate the routers that have gone down, increasing the amount of time necessary for the Network Engineers to locate the routers needing immediate reconfiguration.

4.3 Key Components

This section goes over the utilized components within the dashboards based on goals and limitations stated in section 3.2.

4.3.1 Specifications

The dashboard application requires two main aspects in order to help network engineers see which video routers have poor health conditions: Splunk Database and an IDE to access Splunk API queries through Python. The Splunk Databases harbors all the system logs of all Routers that are under Verizon's authorizations. To develop the dashboard on the Splunk interface, an IDE can be used to develop a script that can access a Splunk account under Verizon's authorization in order to develop queries in a programming language, such as Python.

4.3.2 Challenges

The most arduous task is receiving and adjusting the dashboard to the needs of all network engineers. Network Engineers must have a friendly user interface that will easily show the data they need immediately. Throughout this process, I had to make edits in my code and frontend portion of the dashboard so that the users are able to find the routers that are in immediate need of reconfiguration to ensure that customers will always have reliable access to network services by Verizon. Additionally, the Network Engineers would also prefer to have visualizations of the locations of the routers in need of critical repair.

4.3.2 Solutions

To combat these challenges, I mainly communicated with the entry Network Engineers at Verizon. I asked them for feedback about what information they would prefer to see immediately on the dashboard and other metrics or telemetry information they can use to help them with the configuration process. I developed a map feature in Splunk that displays all routers locations through markers and their Cisco severity levels to indicate which routers are in need of reconfiguration.

4. RESULTS [or ANTICIPATED RESULTS or other title as appropriate]

[Anticipated Results (or Anticipated Outcomes; based on process design concept and, possibly, results reported for similar apps)]

Compared to the Vrepair application, this dashboard has helped Verizon reduce by roughly 30% the amount of time needed to find and locate routers for reconfiguration. This allows Network Engineers to focus on other tasks that can help improve Verizon's network rather than focusing more time towards scrolling through the Vrepair application and checking the hundreds of routers under Verizon's domain.

5. CONCLUSION

[Conclusion (brief summary of project's importance: need, meaningful elements, features, uses, benefits, anticipated value to consumers, etc.)]

The purpose of this project was to develop a dashboard application to mitigate the amount of time used to reconfigure Verizon's routers. The software I developed helped Network Engineers to immediately find routers that are in critical need of reconfiguration to ensure that customers utilizing Verizon's network are having high bandwidth and network. The features included in the dashboard include the interface showcasing routers that are in need of reconfiguration immediately, the location of the routers, and the severity levels and associated errors that caused a need for their reconfiguration. Through developing this dashboard, I have learned how to utilize software tools, such as Splunk, to communicate with programming languages through an Application Program Interface. Additionally, I have learned how to take customer feedback and apply their criticisms to fix any needs to ensure they are able to utilize the dashboard optimally. Overall, the dashboard will help reduce the amount of time necessary for network engineers to locate routers, allowing them to focus on other needs within Verizon to ensure customers are satisfied with its network.

6. FUTURE WORK

[Future Work (next steps needed to complete, expand, launch the app, possibly including other potential uses)]

Future improvements to the dashboard include adding a map interface to help visualize the location of the routers in case natural disasters occur within certain states. Additionally, adding other subqueries within the dashboard to help Network Engineers to find routers with specific traits- Type of Router-in case an Network Engineer wants to examine past logs of a specific router that does not need immediate reconfiguration.

RÉFÉRENCES

[References (list of references cited anywhere in report, including but not limited to Related Work section)]

Network monitoring: A beginner's guide.(n.d.). Splunk. Retrieved October 26, 2022, from https://www.splunk.com/en_us/data-insider/what-is-network-monitoring.html

Saha, S., & Majumdar, A. (2017, October 19). *Data Centre temperature monitoring with ESP8266 based wireless sensor network and cloud* *based dashboard with Real Time Alert System*. IEEE Xplore. Retrieved October 26, 2022, from https://ieeexplore.ieee.org/abstract/document/8073958

Lauriault, T. P., Kitchin, R., & McArdle, G. (2010, December 1). *Knowing and governing cities through urban indicators, City Benchmarking and real-time dashboards*. Knowing and governing cities through urban indicators, city benchmarking and real-time dashboards. Retrieved April 3, 2023, from https://www.tandfonline.com/doi/full/10.1080/21681376.2014.983149

Chen, Y.-J., & Chien, H.-Y. (2017). *IOT-based Green House System with Splunk Data Analysis | IEEE ...* IEEE Explore. Retrieved April 3, 2023, from https://ieeexplore.ieee.org/abstract/document/8256458/

Analysis of the Tay Chatbot Case through Actor Network Theory

A Research Paper submitted to the Department of Engineering and Society

Presented to the Faculty of the School of Engineering and Applied Science

University of Virginia • Charlottesville, Virginia

In Partial Fulfillment of the Requirements for the Degree

Bachelor of Science, School of Engineering

Shiva Manandhar

Spring 2023

Advisor

Benjamin Laugelli, Department of Engineering and Society

Introduction

In 2016, Microsoft introduced Tay, an Artificial Intelligence Chatbot, that was capable of learning and recognizing human text and speech patterns through communicating with users and responding to them on social media platforms, such as Twitter. However, the Chatbot became very hostile and posted hate speech on Twitter due to some users taking advantage of the Chatbot's algorithms, causing Tay to learn negative behavior and language. The Chatbot was spewing racist, sexist, and anti-semitic responses on Twitter to some users. As a result, Microsoft discontinued the Chatbot within 24 hours due to the harmful speech it was saying on the platform (Zemčík, 2020).

The adverse case of Tay demonstrates to researchers how Artificial Intelligence can perform actions in their behaviors that were not intended and the needs for careful moderation and control. Chatbots have proven to be capable of responding and mimicking human speech, and they are used by many corporations to interact with users on technological platforms. But discussion about how Chatbots become hostile has not garnered much attention nor research. There has not been extensive research that explores how and why the algorithms that were utilized by Tay caused the robot to become hostile. There was, however, much research into how adverse users on Twitter eventually lead the Chatbot to become hostile overtime. Due to the public not being informed about other contributions as to why the robot became hostile, many scientists and programmers may not be cautious of the dangers Artificial Intelligence can pose without imposing limitations on its software models. If the public is not aware of the true dangers that can make Artificial Intelligence hostile, it could potentially cause future Artificial Intelligence products to replicate the same adverse actions that has caused Tay to become rogue.

This repetition can be seen when the Lee Luda Chatbot in Korea became rogue through users feeding the Chatbot homophobic slurs, causing the Chatbot to spew homophobic phrases to its customers (Jang, 2021).Further research as to why Tay went rogue can allow individuals to learn more about how to prevent Artificial Intelligence from performing actions that are unintentional and adverse.

Although some people have the conceptions that Chatbots become rogue due to learning data from online users, such as malicious individuals on Twitter, through analyzing the overall network, components, and actors that lead to the negative actions and discontinuation of the Tay Chatbot, I will illuminate the overlooked actors that were responsible towards the hate speech Tay was spewing on Twitter. I will utilize the Research Paper, *Why we should have seen that coming: Comments on Microsoft's Ta*y, to illuminate the overlooked components of Tay that played an important role towards its behavior and shutdown Specifically, I will utilize Actor Network Theory, a framework that analyzes the social and natural world in terms of relationships and networks (Cressman, 2009), to analyze the machine learning algorithm and the Tay Chatbot developers' responsibility that were omitted from discussion and prominent actors for Tay's actions.

Background

Tay was developed by Microsoft Technology, Microsoft Research, and Bing Team in order to study how Artificial Intelligence Chatbots are able to develop deep conversations. Tay's data were collected through mining past public data that facilitated conversations and utilizing Artificial Neural Network and Text and Speech Recognition algorithms to keep track of past conversations the Chatbot had learned in order to develop proper responses. Microsoft designed

Tay to continually learn from previous users it had interact with and mimic their speech and text patterns. Microsoft decided that the Chatbot's main demographics were in the range between 18 and 24 year olds due to the group being the main users who utilize social media (Foley, 2016). On March 23rd, Tay was released for the public to interact and develop conversations. However, many users who were communicating with Tay were supplying the Chatbot with derogatory speech. As a result, the Chatbot began saying racial and sexist responses to users on Twitter in its consecutive responses. Microsoft later took Tay down the following day and issued an apology for not taking into consideration the repercussions that could have occurred. To this day, the Chatbot has not been instantiated.

Literature Review

There are scholarly articles that research into the performance of Tay and how Tay's responses on Twitter changed people's perception on the capabilities of Artificial Intelligence with public use. However, not much research has gone into understanding whether Microsoft has placed limitations on Tay's machine learning algorithm to prevent it from learning negative behavior while it is interacting with Twitter users. Additionally, little research was conducted as to whether Tay was tested to ensure that it was communicating properly with its intended audience, and no "bugs" would occur as a result of training based on the conversations it was facilitating.

Much research has been done to determine how Tay was able to learn from users by studying its efficiency in the backend of its machine learning algorithm and determining its performance in terms of how well of a response it is able to develop. In the research paper, *Intelligence Analysis of Tay Twitter Bot*, the authors Mathur, Stavrakas, and Singh found that even though Tay was online for 16 hours, it was able to perform competently in learning tweets. The researchers determined that Tay was authentic with its response due to its strength in memorizing large amounts of vocabulary through its machine learning algorithm. Additionally, Tay's Bot Intelligence Score, a scoring algorithm that the researchers have devised to determine the intelligence of Chatbots, was at 99.438 and could have increased even more had Tay been on for further usage by the public if limitations were placed to prevent it from becoming hostile (Mathur et al., 2016). While this work does demonstrate that Tay is quite capable of going beyond the power of most bots through providing very intricate responses on Twitter, the authors did not go into detail as to why the Chatbot was delivering such heinous responses that are negative towards the public. They do indeed give insight as to why the Tay bot was able to provide specific responses towards the users, but the authors never researched into why the Chatbot was delivering hateful messages towards certain users on Twitter's platform. Without knowing as to why Tay was showing indignation from its algorithms, there will not be much information about how Tay's algorithm developed such hostile responses or preventions that are able to be applied to the Chatbot to prevent such an act from occurring on future Artificial Intelligence technology.

In the scholarly research, *Talking to Bots: Symbiotic Agency and the Case of Tay*, Gina Neff and Peter Nagy, researchers from the University of Oxford and Arizona State University, discuss the responses that were garnered after Tay was shut down by Microsoft. The authors analyze the history of Tay's behavior, responses, and aftermath of how the 96000 tweets that were made by the Chatbot affected the platform and why the targeted audience did not allow the Chatbot to act in a friendly and constructive manner Microsoft intended (Neff & Nagy, 2016). Additionally, the authors discuss how Tay needs to have better restrictions on what Tay is able to learn from to prevent the Chatbot from developing adverse responses on Twitter's platform. The authors discuss how the targeted age group did not take the responses from the Chatbot due to its sexist and racist remarks. The authors recommended that there should be more provisions needed to ensure that future Chatbots do not become belligerent, but they do not discuss about where the restrictions should be placed- whether the algorithm or the user responses to the Chatbot should be restricted- nor go into the overlaying architecture of the algorithms in Tay. Thus, like Mathur, Stavrakas, and Singh, without understanding the underlying components that were responsible for the aggressive response from Tay, it will not illuminate how future Chatbots need limitations placed within their software to prevent such acts occurring in the future.

The two scholarly articles do not provide insight of Tay's backend that can demonstrate as to why it wrote hate speech on Twitter. Although they do go into detail about the necessities for further limitations on Tay's software, they do not go into detail as to what components of Tay should be limited. Thus, this paper will dive deeper into the overlooked actors that contributed to the Chatbot's adverse behavior.

Conceptual Framework

Understanding the reasoning as to why the Tay Chatbot went rogue can be explained through the lens of the Actor Network Theory (ANT). According to Cressman, ANT is a framework that focuses on the social heterogeneous networks between human and non-human actors that are kept together through a builder to solve and partake in a challenge (Cressman, 2009). ANT was developed in the 1980s by Michel Callon, Madeleine Akrich, and Bruno Latour in order to comprehend the many anomalies within an event or idea that are strung together by a network. Originally, the creators of ANT utilized the framework to describe particular approaches to scientific and technical innovations that occurred. But Cressman advocates that ANT can now be used to illuminate complexities and relationships of our sociotechnical world (Cressman, 2009). ANT mainly focuses on approaching subjects towards the ideology of “science and technology in the making” rather than “ready made science and technology” (Cressman, 2009). It examines the actors that can either be human or non-human that orchestrated certain aspects or results of an underlying network, which can represent an event, object, or ideology. From ANT, one can find which actor is contributing to a certain feature or aspect of a network that may have caused it to act in a certain manner that was either intended or not intended. ANT starts from standard, unconnected localities (which can be the actors) and then develop connections to see how they formulate the overall network (Latour & Welt, 1996) . ANT can be used to comprehend the factors, which are the actors, that contribute towards the specific nature or purpose of a component within an network and lead to a better comprehension of the causation as to why certain non-human or human actors behaved within the network. This can allow individuals who are utilizing this framework to have a better overview as to how certain events contributed towards specific actions within the network to better evaluate root causes that have occurred. Drawing on the ANT framework, in the analysis that follows I begin by demonstrating what actors, both non-human and human, have been overlooked into the overall network of the Tay Chatbot's shutdown to illuminate to the public the changes in the future that can take place to ensure that such implications do not occur.

Analysis

The developers and the overlying structure of the algorithms used within Tay are crucial actors and non-human actors within the network of the research Microsoft was developing towards the creation of Tay. ANT can show the actors that are threatening the stability of Chatbots from not performing well in the eyes of their developers and potentially causing havoc on society through spreading malicious dialog. Through ANT, underlying that there were other factors that were contributing towards Tay's adverse behavior will bring proper attention towards comprehending as to why the Chatbot's network caused it to become rogue, and how the underlying components that lead to its rogueness could potentially be fixed through looking at the human and non-human actors.

Learning Rate of Algorithm

Understanding the reasoning as to how the algorithm behind the functionality of Tay caused it to become rogue can demonstrate that the machine learning algorithms are crucial non-human actors that were overlooked. In the research article, We Should Have Seen That One Coming, Miller and Grodzinky state that Neural Network Algorithms, which are algorithm that are utilized in Tay, do not have restrictions inherent within their algorithms; neural networks change their model dynamically as they gain more input or data from a specific source (Miller & Grodinsky, 2017). Additionally, Artificial Neural Networks (ANN) are a type of machine learning model that intend to mimic the underlying functionality of the human brain by having nodes that represent a specific numerical weight to remember patterns (What are neural networks?, n.d.). ANNs are able to store information they have learned through particular streams of data by changing the weights of particular nodes within the layers to store information that is critical for potential recognition or predictions. Since ANNs are not able to limit which type of data they are able to read in on their own, ANNs are somewhat forced to learn the information that is being streamed to them, whether it is when it is training or delivering an output to a user. Since there are no restrictions within Tay's ANN algorithms according to Miller and Grodzinky, Tay will be forced to learn and replicate the responses that it collects due to its algorithms' nature to maintain what they learn through data collection. This can lead to ANNs within Tay to mimic behaviors that are occurring frequently, such as the multiple adversaries that are feeding Tay inappropriate information through Twitter. This demonstrates that the non-human actor of the ANN algorithm played a huge role in the overall actions of the network of the Tay Chatbot being rouged and getting shut down by Microsoft. Due to users feeding Tay with malicious text messages that were targeted to be misogynistic and racist, the ANN would constantly adapt its neural nodes to respond adversely. Through repeated exposure from adverse groups, Tay would respond in a malicious way to the users on the Twitter platform due to copying users who previously took advantage of the Chatbot's internal algorithms (Vincent, 2016). ANT shows that the main contribution towards the pitfalls of Tay can be seen in the non-human actors of the algorithms within its structure. Without proper restrictions that can prevent the ANN algorithms from reading adverse data, the network that Tay created would always respond similarly to the user groups it had interacted with due to the inherent nature of its ANN algorithms. This will cause it to respond with hostility had a previous group fed it with adverse text on social media.

Developer Contribution

Another fault through the lens of ANT can be seen through the developers of the Tay Chatbot who were capable of ensuring the Chatbot was performing as intended while it was interacting with users. Miller and Grodzinky argue that designers of the Chatbot system should have taken proper care of Tay to ensure that the dialogue it produced was not adverse; testing with multiple diverse users and specific cases that can potentially cause Tay to develop vulgar responses should have been conducted to ensure all the potential scenarios that the Chatbot can encounter will not cause it to become rogue (Miller & Grodzinksy, 2017). Miller and Grodzinksy noted that the developers did not test Tay with test users that would communicate with the Chatbot to see how it would respond to users on social media platforms. Had the developers tested Tay, specifically with examples of harassment that is posted on social media, they would have noticed that the Chatbot is not ready to be launched on a public platform and would need further testing and modifications to ensure it will not be hostile. Through ANT, the developers played a crucial role towards the creation and maintenance of the Tay Chatbot. Although they were able to develop a Chatbot that was indeed able to communicate and establish a form of conversations with individuals, the developers have not heavily tested the potential of certain groups of individuals trying to change the anatomy of the model, causing the Chatbot to change its behavior. Model testing, which is a process that involves in explicit checks for behaviors that developers expect to occur, and model evaluation, which covers metrics and plots that summarize performance on a validation or testing set, are very critical towards ensuring Artificial Intelligence technology functions as properly as they should and discover new edge cases (Jordan, 2020). Had extensive testing and evaluation of the models been conducted for the specific case of certain attackers trying to maliciously change Tay's behavior, the overall network of the Chatbot would have been more stable for it to have cordial and positive conservations on social media platforms. Through the lens of ANT, the developers, one of the main actors of Tay, lack of tests and placing limitations on the Chatbot held responsibility for causing the Chatbot to act belligerent on Twitter.

As I have argued, the Developers should have conducted extensive testing to ensure that the Tay Chatbot was performing as intended. However, Miller and Grodzinky did acknowledge that there was some testing conducted by Microsoft to ensure that the Chatbot did not go robust. Microsoft did perform tests to see the performance of Tay before launching the Chatbot to the public. According to an article written by Peter Lee, corporate Vice President of Microsoft Research, Microsoft has done stress tests on varying conditions that can occur on the Tay Chatbot, as well as done much filtering and extensive user studies to examine the Chatbot interactions with users (Lee, 2016). Some individuals may argue that since the Chatbot was tested effectively by Microsoft then it may not have been the Microsoft developers fault for the Chatbot's adverse behavior, but rather the demographic that Tay Chatbot was learning from. However, Lee states that the reason Chatbot went rogue is due to a "specific attack" of a subset of people (Lee, 2016). Miller and Grodzinky noted that this vulnerability was apparent to the developers of the Chatbot after Tay's dialogue became adverse (Miller & Grodzinky, 2017). The view that the Microsoft developers were not responsible for Tay's actions is flawed due it not considering that Microsoft and Microsoft Research did not test social exchanges that involve malicious users on social media platforms to interact and change Tay's behavior to become adverse. Thus, the developers' failure to consider all possibilities of potential cases, which could have been prevented through filters in Tay's software, contributed to Tay's hate speech on the platform.

Conclusion

Through the lens of Actor Network Theory, the Tay Chatbot case demonstrates that there are notable human and non-human actors that are often overlooked, contributing towards the overall adverse nature of the Chatbot within twenty four hours of its launch. Although there has been extensive testing on the Chatbot to ensure that it would not have gone rogue, Microsoft did not take into consideration of the non-human actor, Neural Networks, which has caused the Chatbot to continuously learns from new responses from users on the platform that causes its algorithms to adapt to the changes due to the usage of the Neural Network Algorithms. Furthermore, the human actors of the researchers who were not aware of a potential group of malicious users have also led them not to develop limitations nor filters beforehand that are able to potentially ignore such an attack by a particular adverse group of individuals. Even though the algorithms have been extensively tested before hand and the Chatbot has been introduced to wide range of individuals of different backgrounds, these actors have lead to the potential fallout of the network of the research underlying how the Chatbot interacts with users would lead to the Chatbot becoming rogue and thus developing hate speech within the Twitter platform. Future developers of artificial intelligence need to be more aware of placing restrictions to ensure that artificial intelligence does behave with right intentions and perform tests that are able to take care of all potential edge cases-from both specific malicious target groups and benign groups. If the public does not focus and understand the other implications that can cause Artificial Intelligence to not perform as intended by developers, then it will cause future software, tools, or

Chatbots that rely on Artificial Intelligence to experience the same issues that Tay faced. If the public knows that the algorithms and developers were at fault alongside the adverse groups that facilitated Tay to become malicious, then they will realize that Chatbots will always interact with some form of adverse users, thus there should be more limitations placed that filters the Chatbot to not become adverse. Understanding the overlooked actors encompassing the Tay Chatbot's downfall can better lead to more secure and proper testing techniques that are able to allow artificial intelligence technology to become much more effective at their particular tasks and prevent any edge cases that would cause them to be misguided severely by consumer use.

Word Count: 3421

References

Cressman, D. (2009, April). A brief overview of actor-network theory: Punctualization, Heterogeneous Engineering & Translation. *Simon Fraser University*. Retrieved February 27, 2023, from https://summit.sfu.ca/item/13593

Foley, M. J. (2016, March 23). Microsoft launches AI Chat Bot, tay.ai. *ZDNET*. Retrieved February 27, 2023, from https://www.zdnet.com/article/microsoft-launches-ai-chat-bot-tay-ai/

Jang, H. (2021, April 2). A South Korean chatbot scandal shows the threat A.I. Presents to Privacy. *Slate Magazine*. Retrieved April 12, 2023, from https://slate.com/technology/2021/04/scatterlab-lee-luda-chatbot-kakaotalk-ai-privacy.html

Jordan, J. (2020, November 12). Effective testing for machine learning systems. *Jeremy Jordan*. Retrieved February 27, 2023, from https://www.jeremyjordan.me/testing-ml/

Latour, B., & Welt, S. (1996). On actor-network theory: a few clarifications. *JSTOR*. Retrieved March 1, 2023, from https://www.jstor.org/stable/pdf/40878163.pdf?casa_token=L222XAgBNGUAAAAA:b6aNM10ArShxlaYgO7MnCOIKvJUzcEM_PVabCjLggyq4ievdi-VMfiiiJU1At2GkcKdl93J4FxHkAxM0IVGkxvq96p_xrcfOYG0CdfRE4aYg2U-pZPo

Lee, P. (2016, March 25). Learning from Tay's introduction. *The Official Microsoft Blog*. Retrieved February 27, 2023, from https://blogs.microsoft.com/blog/2016/03/25/learning-tays-introduction/

Mathur, V., Stavrakas, Y., & Singh, S. (2016, December). Intelligence Analysis of Tay Twitter Bot - IEEE Xplore. *IEEE Xplore* . Retrieved February 27, 2023, from https://ieeexplore.ieee.org/abstract/document/7917966

Miller, M. J. W. K., & Grodzinksy, F. S. (2017, September 1). Why we should have seen that coming: Comments on Microsoft's Tay. *ACM Digital Library*. Retrieved February 27, 2023, from https://dl.acm.org/doi/pdf/10.1145/3144592.3144598

Neff, G., & Nagy, P. (2016). Talking to Bots: Symbiotic Agency and the Case of Tay. *International Journal of Communication*. Retrieved February 27, 2023, from https://ora.ox.ac.uk/objects/uuid:613f7303-8a07-4f5a-ada2-b495c9a449af/download_file?file_format=pdf&safe_filename=Neff_Nagy_2016_Talking%2BTo%2BBots.pdf&type_of_work=Journal+article

Vincent, J. (2016, March 24). Twitter taught Microsoft's AI chatbot to be a racist asshole in less than a Day. *The Verge*. Retrieved February 27, 2023, from https://www.theverge.com/2016/3/24/11297050/tay-microsoft-chatbot-racist

What are neural networks? (n.d.). *IBM*. Retrieved February 27, 2023, from https://www.ibm.com/topics/neural-networks

Zemčík, T. (2020, September 2). Failure of chatbot tay was evil, ugliness and uselessness in its nature or do we judge it through cognitive shortcuts and biases? - ai & society. *SpringerLink*. Retrieved February 27, 2023, from https://link.springer.com/article/10.1007/s00146-020-01053-4

Redesign of the Computer Science Department for real time data collection and user feedback.

Analysis of failures for preparing students in the UVA's Computer Science Department

A Thesis Prospectus
In STS 4500
Presented to
The Faculty of the
School of Engineering and Applied Science
University of Virginia
In Partial Fulfillment of the Requirements for the Degree
Bachelor of Science in Computer Science

By

Shiva Manandhar

October 27th, 2022

Technical Team Members:
Shiva Manandhar

On my honor as a University student, I have neither given nor received unauthorized aid on this assignment as defined by the Honor Guidelines for Thesis-Related Assignments.

ADVISORS
Benjamin Laugelli, Department of Engineering and Society
Brianna Morrison, Department of Computer Science

Introduction

Developing dashboards for trends and real time data has become an important skill in the Computer Science field to see trends and patterns in real time. Network Dashboards, for example, are constantly monitoring the health of networks by senior level professionals at any time due to their uptime in collecting real time data (Saha & Majumdor, 2017). University of Virginia (UVA) Computer Science courses have demonstrated a wide array of web design and software development in their content. However, without emphasizing the teachings to retrieve real time data through a specific database and communicating what should be displayed on developing software or dashboards from the users, it can lead to little information that is viable for users to comprehend. Thus, dashboards with real time data that meet the criteria needed by users who are using them must follow these logistics to ensure that it showcases the intended information necessary for the consumers. To develop software interfaces that display real time data, UVA Computer Science courses need to delve further into the process of teaching how to collect real time data and enhance students' knowledge of communication with potential consumers to avoid errors in the development process.It is important to understand the social and technical applications of not having the capability of utilizing well known software that is able to collect real time data for students to have a better knowledge of how to collect it in order to better apply these technologies to the real world and their jobs and meet customer demands. In the technical aspects, without having the knowledge to capture data, it would be difficult for engineers to utilize software tools to help them display the data on applications and projects. In terms of non-technical and social factors, not being able to show what is necessary and meet the demands of what consumers want to see can be problematic for developers who are trying to meet the demands needed to be successful in their jobs. Both ideas are needed to understand the implications that can be caused that can mitigate the chances of new graduate engineers being successful in their careers. In the Technical Proposal, I will explore how the implications of not teaching software tools and collection of real time data can be problematic for new graduating Computer Science students when they are entering the workforce through examining ways and a case study to mitigate this pertaining issue. In the STS Proposal, I will examine the missing components in the network of the Computer Science department to demonstrate how it can be fatal for the future graduates of UVA's Computer Science department in tackling challenges that can occur in their work and projects and not allow them to be better prepared for the labor market.

Technical Project Proposal

In June of 2022, I had the opportunity to intern at Verizon and work on creating a dashboard through database tools and languages, such as Splunk and Python. The dashboard I was tasked to develop required recording when real time outages and network messages from speed test routers occurred so that network engineers are able to configure these routers that are damaged. The dashboard was meant to be used by network engineers to debug routers in a designated interface when they are down in the dashboard. However, I came ill prepared with developing such a dashboard; I was not familiar with collecting real time data through Splunk and continually improving the dashboard through asking for feedback from the network engineers on what parameters need to be displayed for them to utilize. Even though Splunk is similar to syntax of programming languages I have used in my courses, I was not familiar with its network monitoring capabilities, which enable companies, such as Verizon, to monitor networks for any potential warnings through its streaming software (*Network monitoring: A beginner's guide,* n.d.).

Due to my lack of understanding of how to communicate with users and collecting real time data, the Computer Science department needs to expound upon how to receive feedback and learn how to display real time data. Although current Computer Science courses in UVA do discuss the necessary software that is needed to develop the dashboards, they do not emphasize the necessities and ways of collecting the data in real time through real time software tools. A study conducted by Radermacher and Walia surveyed 23 industry professionals and determined that most recent Computer Science Graduate students seemed to not have the knowledge of software tools (Valstar et al., 2020). This demonstrates it can be problematic that recent graduates do not have the necessary tools to be able to develop software when they enter the workforce, demonstrating that they are unable to utilize important software development tools to extract and communicate information towards the development of software and dashboards. Students also lack the ability to create a plan of execution of what their software can potentially look like. According to a study conducted by researchers in Appalachian State University, 55 % of college students do not develop a plan before coding (Norris et al., 2008). Because these students just go straight into the coding process, they do not fathom whether they must take into consideration what the customers want from the specific software that can be displaying real time data, which could mitigate the successfulness of the software's capabilities, relating to how education fails to prepare students with a plan of execution to develop software. Current courses in UVA do go over important tools that are critical towards software development. For example, CS 3240 Advance Software Development does indeed go over important software methodologies that are needed in order to develop proper software for human use(Sheriff & McBurney, n.d.). Even though there is a course that does delve into the details of developing software, which is critical towards developing software applications and dashboards, it does not go into ways to make the software update with real time statistics of current events occurring, which can be necessary for students to create in industry. However, there are research groups in UVA that delve into the focus of collecting real time data and displaying it through software applications. Floodwatch, a research project that aims at collecting weather data in Asian countries to forecast weather conditions and hazardous events, such as floods, is developing software that makes future forecasts and predictions for users through their app (Nguyen, 2022). The research group in UVA uses real time weather data to display to users in Southeast Asia through utilizing software engineering principles and gather real time data through network technology that is stored on the database, which is then seen on the application. This demonstrates the need for projects such as the software application being deployed and conducted by Flood Watch to be taught to students in the UVA Computer Science Department in order to make them more adept at developing software and asking for feedback from customers.

In my technical report, I will discuss methodologies of how the current Computer Science curriculum can improve student learning through collecting real time data and learning how to apply them to actual software and how to get feedback from individuals utilizing their software. I will develop surveys of what courses that UVA teaches that allow them to have a better comprehension of data collections, whether they have been taught software and tools that are used in jobs. I will survey alumni of Computer Science students at UVA who are currently in an industry Computer Science field and ask what they wished the University had taught them to make them more prepared for their role. Furthermore, I will also explore a framework that examines teaching methodologies employed in other universities that can allow students to learn more about software tools such as a case study done in the Software Design course. In Dartmouth University, students were exposed to a non-traditional taught Computer Science course where they worked with a diverse set of tools and software that allows them to work on multiple projects relating to software and machine learning development (Linder et al., n.d.). The techniques proposed to teach these students facilitated them to become more comfortable with software tools that are commonly used in software and artificial engineering jobs, thus allowing these students to be more prepared for their future endeavors in the workforce Through examining the techniques proposed in this paper, I will be able to propose more projects that encourage and teach students to be better adept at using software tools, allowing them to be more prepared for the industry.

STS Project Proposal

There are some components of UVA Computer Science department's that are missing that contribute towards the lack of feedback and real time data collection that students currently have in its system. Currently, UVA's Computer Science Department lacks the necessary ability to facilitate individuals in the department with the knowledge to collect and utilize software that is able to display real time data. Although the department of Computer Science is able to produce a large number of students that are able to enter the workforce immediately after their graduation, they are unable to teach these students the necessary ability to collect real time data and utilize software that is commonly used in the industry, causing them to potentially not meet expectations for them in their work. If we examine the actors of software technologies that are able to collect real time data, then we are able to see a small fault in an already successful network that the Computer Science Department has, to enable students to have a better comprehension of how to collect data in real time. Through Actor Network Theory, (ANT) the current network that the UVA department of Computer Science has built does indeed prepare students for some level of awareness and expertise in their field of study, but it is apparent that there is a weakness in within the network that is not making new graduate students as prepared and knowledgeable about the tools in the software development workforce due to the social actors of customer feedback and technical actors of software tools not being present in the network. The framework used in this paper, called ANT , is a framework that focuses on the social heterogeneous networks between human and non-human actors that are kept together through a builder to solve and partake in a challenge (Cressman, 2009).

The Computer Science department for its bachelor science program strives to ensure that students are able to make tangible contributions towards their profession through being innovators in design, analysis, and application of computer systems in their courses (*Computer Science (B.S.),* n.d.). Some argue in the department of Computer Science that the reason as to why some new graduates may struggle in their new careers is due to the lack of knowledge of algorithms and development that is needed in order to be successful in the software engineering industry. However, the current Computer Science curriculum has many courses through its teachings of algorithms, software engineering, and computational theory to ensure that students are indeed well versed in the analysis of programming (Tychonievich & Sheriff, 2022). The curriculum does indeed prepare students for the most essential principles that are needed to be successful in their prospective careers relating to Computer Science. This can be seen through UVA's ranking in Code Signal since it is the top ranking school in Code Signal's General Coding Assessment, which are tests that are used to indict new graduates into successful careers at prominent companies, such as Uber, Meta, and Reddit (McManamay & Kelly, 2022). This demonstrates that UVA is able to garner many of its graduates to get into very prestigious companies for software engineering; however, there are some components that are overlooked and currently lacking that can inhibit students from being able to ensure that the software they are developing is indeed helpful towards their users. One of the most important courses for data collection in the current Computer Science curriculum is Databases. The Database course in UVA does go into detail of how to create and potentially utilize data in their software, but it does not focus on how to obtain the data real time or peer reviewed by potential users of the software (Praphamontripong, 2022). Furthermore, the software developed does not get customer feedback, which is a critical part of the network that is missing for students to succeed in software development. Human Computer Interaction, another computer science course in the department, goes over user/task analysis and prototyping for the software development process (Horton, n.d.). Even though the class does indeed go over the essentials for making software more friendly for users, it does not employ an emphasis on getting feedback from potential users throughout the course. This can cause software developers from UVA not to have the ability to further progress software that they are developing to meet all the needs for the customer, thus potentially making the customer experience while utilizing the software not as user friendly or missing some key components that are necessary for them to succeed. Even though UVA's department of Computer Science facilitates an environment that is able to allow students to be successful in working and getting into industry jobs, the network overlooks the weakness through a social lens of not including actors and non-actors, such as customer and customer feedback, that many students need when entering into the workforce. From a technical lens, not being taught how to utilize software tools that can collect real time data can also be catastrophic towards the success of a new graduate student in the workforce. This demonstrates important components the UVA Computer Science department needs inorder to ensure students do indeed have a well professional background for success in their prospective career.

Through ANT, it is apparent that there are many actors and nonactors involved in UVA's current curriculum, such as software development, data structures, and users/usability. Even though the Department of Computer Science has established the use of such actors of software development through their classes, they need to expound upon receiving critique from an expanded audience. Learning how to get and employ the feedback gained from customers is a critical non-actor missing in UVA's network adding; the additional elements of teaching how to survey the demographic needs of their potential clients/users of software can help students develop software that will benefit the user even more through receiving feedback from them. Customers or users is another actor not present in the network, making it difficult to meet the expectations that they need while not giving them the opportunity to review the software. Furthermore, they would need the non-actor of collecting real time data; if courses that already focus on collecting data, then allowing them to utilize tools, such as Splunk, that can capture data would make them better at developing software that is more adept at showing trends and incur tasks needed. To support my stance through the framework of ANT, I will find studies relating how to teach software development techniques for displaying real time data from other universities and research teaching methods for how to properly use software to display data. I will gather student feedback from surveys that will ask them if they received feedback from software or websites they develop in class and whether they are taught how to capture real time data.

Conclusion

The reformation of how to improve the Computer Science curriculum to better suit those needs of students for software feedback and collection of correct data in a well presented manner will be beneficial for students' careers after graduating. The department could enact a new course that focuses on the development process of creating software while having public individuals critique their software throughout the software development process. Additionally, there could be a Computer Science course that is related to storing and retrieving data that is collected in real time to enable students the ability and experience to display real time data for customers and/or workers within their corporation to have a better understanding of what critical trends are occurring within their business. The STS Proposal can help assist me in the Technical Project through the actors that are contributing towards the lack of tools and knowledge that students are not fully prepared for to do well in their jobs. Through the knowledge of the actors from ANT that contribute to the success and weakness of the UVA Computer Science Department's Network, I can better employ the design methods from the Technical Project through pinpointing the specific actors and non-actors that are needed for the graduates going into their full time positions. The Technical Project contributes towards the socio challenge that new graduate students face in industry through providing implications and methodologies to improve their overall knowledge and capabilities of software tools and data collecting in the department. Additionally, the STS Project will give us a thorough understanding of what missing components or actors are needed for students to better prepare for their jobs in terms of the usage of software tools, allowing the Department of Computer Science to change and implement new teachings to better help those students in their future careers.

References

Computer Science (B.S.). (n.d). Program: Computer Science (B.S.) - University of Virginia - Acalog ACMS™. Retrieved October 26, 2022, from http://records.ureg.virginia.edu/preview_program.php?catoid=49&poid=6227

Cressman, D. (2009, April). *A brief overview of actor-network theory: Punctualization, Heterogeneous Engineering & Translation*. Simon Fraser University. Retrieved December 1, 2022, from https://summit.sfu.ca/item/13593

Horton, T. B. (n.d.). *CS 3205 – HCI in software development - University of Virginia school* University of Virginia. Retrieved October 27, 2022, from https://www.cs.virginia.edu/~horton/cs3205/cs3205-1-intro-f16.pdf

Linder, S. P., Abbott, D., & Fromberger, M. J. (n.d.). *An Instructional Scaffolding Approach to Teaching Software Design*. Dartmouth College. Retrieved December 7, 2022, from https://d1wqtxts1xzle7.cloudfront.net/36083640/312-libre.pdf?1419830790=&response-content-disposition=inline%3B+filename%3DAN_INSTRUCTIONAL_SCAFFOLDING_APPROACH_TO.pdf&Expires=1670468195&Signature=SzL0MH4qnUBKJa4p8RrHwnMQnBnfpfUV5Dde3r6O1Amuw69Tu7femBLhTVVVtKD4ms7S1SolN2VqB1LJJ-T7A2EPEXkkp8lRFTaIylnvJg4UpM2nF6gWK6cGko6zum~tiB1fYalCzy-dLRlQfwczgBGdsXV5IACrvngBiZZJqoP9I1DOM8Tv4lsrgSpg92PSiKPvmqSLY1N~wdPS1l7NkXrEz~vmQT1yYcjfqhL6SjCSnoZLzXTO3QeBZsaFN1uq9mngDQe4EOXZRhAbGL7pV8v3egj

P5i21AjSDg2ndadGIK0m93tuJBD9vc2L7FX8zrziEohJS46Er-E8KxZyOLg__&Key-Pair-Id=APKAJLOHF5GGSLRBV4ZA

McManamay, J., & Kelly, J. (2022, June 14). *UVA no. 1 in software engineering, topping Standard Bearers Stanford, UC Berkeley*. University of Virginia. Retrieved December 7, 2022, from https://engineering.virginia.edu/news/2022/06/uva-no-1-software-engineering-topping-standard-bearers-stanford-uc-berkeley

Network monitoring: A beginner's guide. (n.d.). Splunk. Retrieved October 26, 2022, from https://www.splunk.com/en_us/data-insider/what-is-network-monitoring.html

Nguyen, N. R. (2022). *Rich Nguyen | Research*. N.RichNguyen. Retrieved October 26, 2022, from https://www.cs.virginia.edu/~nn4pj/research

Norris , C., Berry, F., Fenick, J. B., Reid, K., & Rountree, J. (2008, June 1). *Clockit: Proceedings of the 13th Annual Conference on Innovation and Technology in computer science education*. ACM Conferences. Retrieved October 26, 2022, from https://dl.acm.org/doi/pdf/10.1145/1384271.1384284

Praphamontripong, U. (2022). *Fall 2022- Syllabus*. University of Virginia. Retrieved October 26, 2022, from https://www.cs.virginia.edu/~up3f/cs4750/syllabus.html

Saha, S., & Majumdar, A. (2017, October 19). *Data Centre temperature monitoring with ESP8266 based wireless sensor network and cloud based dashboard with Real Time Alert System*. IEEE Xplore. Retrieved October 26, 2022, from https://ieeexplore.ieee.org/abstract/document/8073958

Sheriff, M., & McBurney, P. (n.d.). *Syllabus*. University of Virginia. Retrieved October 26, 2022, from https://f22.cs3240.org/syllabus.html

Tychonievich , L., & Sheriff, M. (2022, February 1). *Engineering a complete curriculum overhaul: Proceedings of the 53rd ACM technical symposium on computer science education v. 1*. ACM Conferences. Retrieved October 26, 2022, from https://dl.acm.org/doi/10.1145/3478431.3499287

Valstar, S., Griswold, W. G., Porter, L., Krause-Levy, S., & Sih, C. (2020, August 1). *A quantitative study of faculty views on the goals of an undergraduate CS program and preparing students for Industry: Proceedings of the 2020 ACM Conference on International Computing Education Research*. ACM Conferences. Retrieved October 26, 2022, from https://dl.acm.org/doi/10.1145/3372782.3406277