ICSI499 Capstone Project Report

Modular Cascade Framework for Statistical Inference

Project Team

Alex Barry (001511254)
Blake Funderburke (001488238)
John Syracuse (001466710)
.....
College of Engineering and Applied Sciences
University at Albany, SUNY

Project Sponsor

Abram Magner
University at Albany, SUNY Department of Computer Science

05-06-2024

Acknowledgements

Our team extends its heartfelt appreciation to Professor Abram Magner for generously sponsoring this project, providing invaluable solutions and ideas, and contributing to its development. Without their unwavering support, this project would not have been possible.

Our team also wishes to recognize Professor Pradeep K. Atrey for his dedication and contributions to this course.

Abstract

Infection data can be used on social media websites to view the data is trending. This can be used to track how hashtags are used and how they spread. For example a Node in this system would be a user who tweeted something using a particular hashtag. Any user who then clicked on the hashtag is now "infected". These infection events are then put into a cascade. Cascading in computer science refers to the process of one operation triggering a series of related operations.

Our team created a program that reads in social media data and is able to detect the infection events. Then, it produced a visualization of the cascade and showed how nodes infected one another over time. Along with the time series data of the infection events the user is able to see how information spreads and when something begins to trend. We were also able to write a program that pulls data from Facebook pages and the user can analyze the data based on keyword, timestamp, and other post information.

There are very few existing solutions to the problem. Mainly because the data has rarely been analyzed and put into a cascade. Some existing cascade solutions include AECasN (AutoEncoder Cascade Networks). "AECasN is an end-to-end predictor, which takes the cascade networks as input and the final size of information cascade as output. It can capture the structural characteristics and the dynamic features of whole cascade networks to map the network instances into low-dimensional vectors, and then predicts the size of the cascade by a MLP (multi-layer perception) component". [1]

Our team faced issues when attempting to construct the cascades and implement the time series data and create a slider. This took research and lots of trial and error. One of the other issues is Facebook API not giving our program permission to pull data without a verified business license. We were able to overcome this by testing the code to pull data on a personal Facebook account. And the user will need to use their own license on a case to case basis.

Contents

1 Problem Analysis

- Data Gathering and a Modular Cascade Framework for Statistical Inference involves learning parameters of a cascade model from infection data. The goal of this project is to develop a extensible, flexible software framework that can pull from a variety of sources and output cascade data.

- This is an important problem because Cascades on networks are used to reflect the spread of information in many different contexts. For example, it can be used to model the usage of a certain hashtag on twitter. Or how an infection spreads across a network of people. This modeling of data in cascade graphs could lead to novel insights into everyday processes

- This problems challenges are related to the implementation of the plugins and the specific criteria required for each, given the varied nature of the datasets associated with a particular plugin. Additionally, the development of a modular framework capable of housing and creating plugins targeting differing infection criteria provides a layer of difficulty. The creation and optimization of a database housing the time series of individual infection instances as well as the ease of analyzing and manipulating said data is another primary challenge that our team will have to overcome.

- AEcasN (AutoEncoder Cascade Networks is one of the few things that is a solution. It is a cascade that is used for predictive analysis of data. "With the emergence of online social platforms, users can post various information and put forward with some of their own ideas on some topics. These activities have promoted the rapid dissemination of information to a certain extent, leading to and stimulating the emergence of information cascade. This phenomenon is very common and can be found in other scenarios except online post forwarding, such as paper citation". [1] AEcasN's main goal is to develop cascades to solve this.

- The reason that this is not a sufficient solution to our problem is because it is not user friendly. The system is difficult to use and hard to decipher what the graphs mean. Another issue with the system is is that different scenarios may require different features or combinations of features meaning the system is not universal and robust.

- The cascade model serves as the foundation for understanding how infections spread through a network. By breaking down the model into modular components, the framework can adapt to different types of infections and data sources.The framework must be capable of gathering data from diverse sources such as research institutions, and social media platforms. This flexibility ensures that the model can be applied to a wide range of infection events. With this we aim to provide a general cascade interface with visualization so that users can quickly prototype and work with cascades.

- Some of the key characteristics of our project are scalability, performance, security and portability. The application should allow the user to successfully add plugins customized to assess infection data of a given graph and produce a time series of said data stored in a database. The application should reliability to be able to query the database for the time series of infection and produce results in no longer than 5 seconds. The application and associated database should not be vulnerable to commonly known and used SQL Injections. The methods related to the application will be available to other prospective users in the form of a Python API.
- Our solution is better than others because ours is the only one specifically designed for this implementation. Others have been used in this way but are not efficient in the area.
- Creates graphs that show how and when nodes become infected.
- Pulls data from social media sources to be analyzed.
- Provides a general interface with visualization so that users can quickly prototype and work with cascades.

2 Proposed System/Application/Study

2.1 Overview

Data Gathering and a Modular Cascade Framework for Statistical Inference that involves learning parameters of a cascade model from infection data. The goal of this project is to develop an extensible, flexible software framework that can pull from a variety of sources and output cascade data. This is going to help users understand how data is being spread and how they can take advantage of this information. This report will be comprised of 7 different sections. Section 1 (above) consists of the problem the project is constructed around and gives the reader info regarding the project. Section 2 will explain the system we are proposing to create the solution to the problem. Section 3 is comprised of the testing done on the project. Section 4 Discusses the Legal and Ethical Practices used in the project. Section 5 shows the effort sharing among the group members. Section 6 will be the conclusion and next steps followed by the references in section 7.

2.2 Project Requirements

In our project the User class is mainly who ever has or needs data that they need to be analyzed and put into a cascade. This could be used in a wide variety of scenarios including possible marketing endeavors as well as academic research. The first functional requirement is to be

able to add customized plugins to the application. One of the goals of our application is to create a robust system that can be used in several different types implementations. So, the user should be able to create their own customizable method for extraction of infection data that can be added to the application directly. The next functional requirement is to be able to construct a time series of a network. The reason this is crucial to the project is because knowing when nodes become infected and when the initial infection event occurs helps the user know when something started trending and it shows the exponential spread of the information. The third functional requirement is to be able to Query social media sites and applications for infection data. The reason for this is the user should be able to pull data directly from the social media website into the program and then be able to make the cascade from that data. Our final functional requirement is to be able to Visualize/Report Data at Desired Time. The requested time series should be able to be visualized, detailing currently infected nodes and nodes most recently infected since the previous instance of time. This is crucial to the project as knowing what the data looked like at a specific time is very important as this will show the differences in how the infections change over time. The first of out non-functional requirement is scalability. The application should allow the user to successfully add plugins customized to assess infection data of a given graph and produce a time series of said data stored in a database. This once again helps our goal of a robust application as being able to handle a data set of many different sizes. Our second non-functional requirement is performance. The application should reliability to be able to query the database for the time series of infection and produce results in no longer than 5 seconds. This is a very important requirement as if the application takes too long to work no one will want to use it, and the application will be rendered useless. The third functional requirement is security. The application and associated database should not be vulnerable to commonly known and used SQL Injections. If the application is not safe to use it does not matter how effective the application is. If the data can be stolen everything that the application can do does not matter. The final non-functional requirement is portability. The methods related to the application will be available to other prospective users in the form of a Python API.

2.3 Technical Design

Our technical solutions to the problems applied to all of our functional requirements. The first requirement was Interface for Plugins. To make plugins truly powerful and versatile, the application needs to give them access to itself, by means of exposing an API the plugins can use. Users must be able to use the library to work with their intended data set. To fulfill this requirement. There were two proposed solutions. Proposed Solution 1: Using Hooks to create the Plugins. A hook is a place in an application where plugins can add functionality. A hook can be used to notify plugins of some application event, or to request plugins participation in some process. Hooks can also just be used to register and find objects that provide some application-specific interface. Hooks are created using plugins.hooks. Proposed Solution 2: Creating the plugins by using classes. One way of creating plugins can be done by

simply creating classes that provide the functions needed for the plugin to work successfully. Ultimately we decided to make create the plugins by using classes. This is the more simple and easy to implement solution. None of the plugins we will be designing need to be too complex or use the hook. The next functional requirement was to construct a Time Series of a Network. The time series of infection for the given network should be constructed using the desired plugin and stored within a database for ease of extraction. The proposed solution is Use the pandas' functions. We will leverage the powerful time-series functionality made available by how Pandas represents dates, by the DateTimeIndex. Python has a wide range of functions that can be used to manipulate the time series data. This is the only solution that is viable and something is our proposed solution as it is the only realistic solution to our problem. The next functional requirement was figuring out a way to visualize data. Here we used a combination of two solutions to solve our issue. NetworkX is a Python package for the creation, manipulation, and study of the structure, dynamics, and functions of complex networks. Nodes on NetworkX can be anything. (Text, images, records). This is perfect for our system. Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. This allows for interactive figures that can be updated. We used both solutions as both Mathplotlib and NetworkX offered many benefits to us. And Finally the last functioal requirement was being able to pull data from social media platforms. The solution we chose was to use the Facebook Graph API to generate an access code and be able to pull data directly from the Facebook page the user desires.

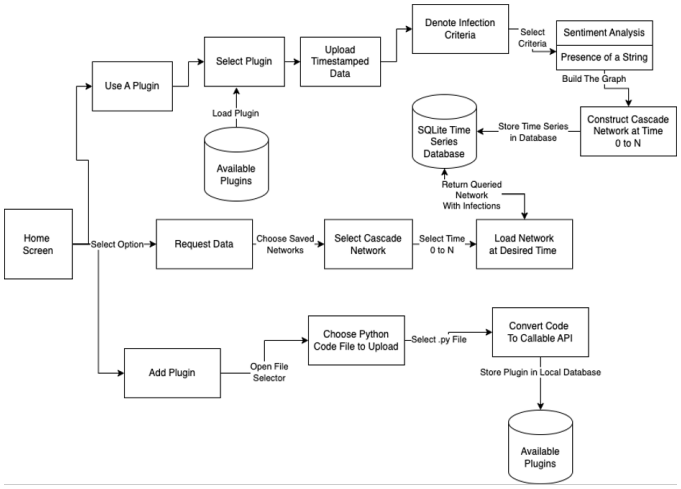

Figure 1: System flow diagram

2.4 System Implementation

For our project all of our work was done in python. The reason for this (along with it being the strong recommendation of our sponsor) was due to how well python does when it comes to

dealing with data. We used certain python libraries such as pandas, matplotlib and NetworkX to help us handle and visualize the data. For version control we used GitHub and all cloned into the repository, made changes locally and then commit and pushed the changes into the repository. We all used Visual Studio Code as our IDE and connected directly to GitHub.

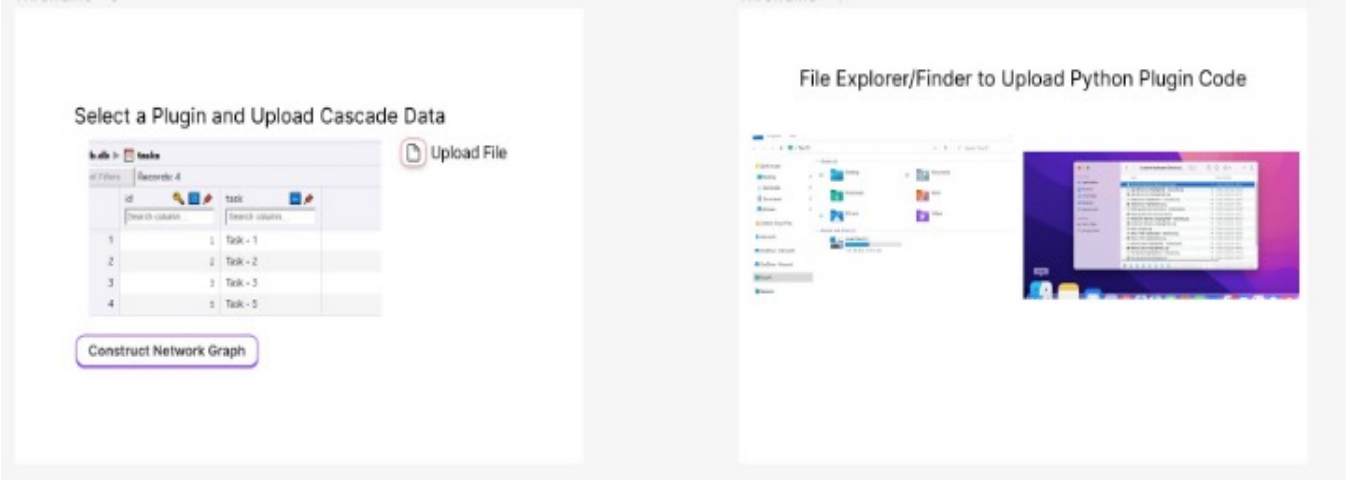

Figure 2: GUI

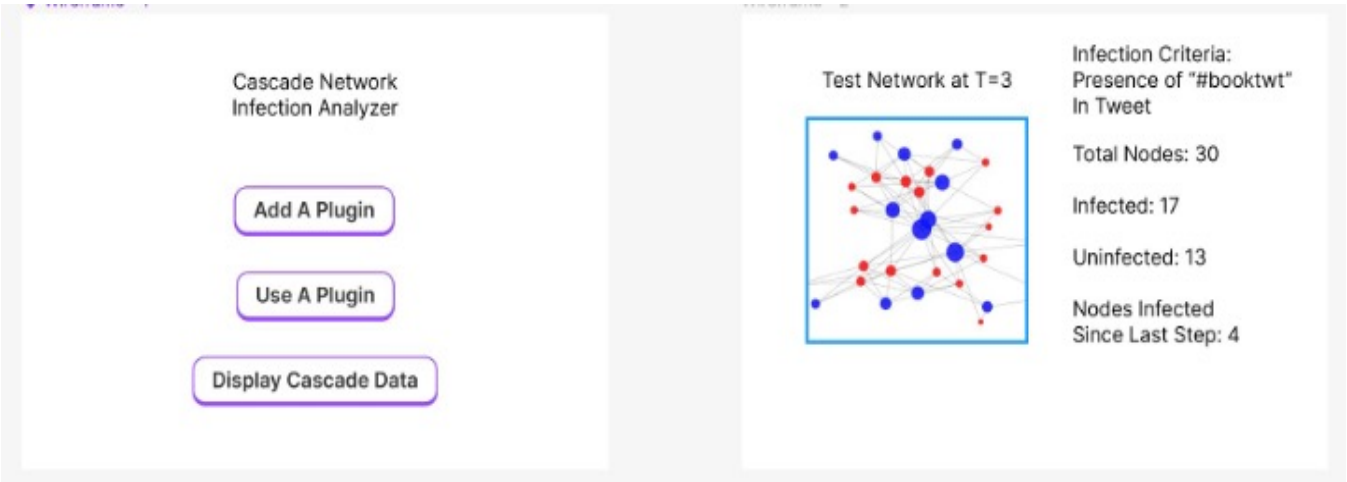

Figure 3: GUI and Visualization of Data

2.5 Use of Computer Science Theory and Software Development Fundamentals

2.5.1 Use of Computer Science Theories

Data Visualization Theory. Data visualization is the graphical representation of information and data. This is done by using elements like charts, graphs and maps data visualization tools provide an accessible way to see and understand trends, outliers, and patterns in data. [2]. This was clear and apparent in our project. Every time we had to create a graph of infection data we had to apply the concepts of data visualization theory. We had to ensure that our graphs were easy to read and could be understand very quickly after looking at them while also maintaining the complexity of the project. The main aspect of data visualization theory that we used was graphs. "A graph is a diagram of points, lines, segments curves, or areas that represents certain variables in comparison to each other" [2].

Graph Theory. Graph Theory was one of the main components of our project. A network is a collection of entities, such as Facebook Users and the connections between them can be represented using a graph [3]. This is exactly what we are doing in our project, as we are pulling data from social media sites and analyzing the infections and displaying them in a graph. On our graph, the nodes are the users themselves and the social media account associated with it. The edges are the connections between those accounts.

Network Theory. This is a theory that involves how different elements in the same network interact with one another [4]. The best way to consider this is to think of the nodes as the things that are connected via the edges. The study of this phenomenon is known as graph theory. This is applicable to our project as we are applying network theory to social networking applications. The Nodes are users, the edges are how they are connected and the network itself is the use of a certain hashtag or whatever the infection event is.

2.5.2 Use of Software Development Fundamentals

Testing. This is one of the most important aspects of software development. Code testing is virtually as important as writing the code itself. This is because it needs to be verified that the code works as intended and the user receives the expected output. [5] This was something that applied to our project countless times. We used pytest to verify that our code gave us the expected. We also had some example integration tests that were written to verify how the graphs worked.

Version Control. This fundamental of Software Development can also be called source control. This is the tracking and managing changes to software code. [6] The form of version

control we used was GitHub. This was essential to our project as we were able to make changes and upload them to GitHub allowing us to work autonomously. This also kept track of modifications made so we can go back and find something that could have resulted in an error after the most recent push.

Object Oriented Programming. This is a computer programming model that organizes software design around data, or objects rather than functions an objects. [7] This was used in our project when we created an interface that created a general way to create a cascade plug in. Users are able to make a cascade that is tailored to their needs based on this. Also, with the language we used in this being Python it was much easier to implement, as python itself is an object oriented language. Some of the benefits that object oriented programming let us implement is making our program much more scalable, modular and polymorphic as it allows to adapt classes to our particular needs.

3 Experimental Design and Testing

In this section, we will describe the testing that we have done to ensure our application works as intended.

3.1 Experimental Setup

- The objectives of our experiments is to be able to produce a cascade from a data set of social media information. We tested to verify that the graphs are accurate and easy to read.
- Our first experiment was a test cascade. This test file creates a new graph with infections. It creates a sample cascade that creates a blue node if it susceptible to being infected. The node is made red if it is infected, and the node is made green if it is recovered.
- Our second test is a test graph. This is a test that makes a sample graph.
- To test the method that pulls data from Facebook pages presented a unique challenge. In order to use the Facebook Graph's API methods pages_show_list, pages_read_engagement, and pages_read_user_content, the application needs to be submitted for review by Meta for Developers. The issue with this is in order for Meta to review your application you need to submit proof of a verified business, which we do not have. Not to mention under the academic research applications the Facebook Graph API methods pages_show_list, pages_read_engagement, and pages_read_user_content are not even available to add. So, the solution we came up with was to create our own page and pull data from that as we did not need permissions to access our own pages page_id.

- We used pytest when testing all of our programs. The pytest framework allows for small, readable tests that are scalable and works well with complex functional testing. [8] Pytest allowed us to test our code as we progressed in the project, as rather than having to test a file once it was complete we were able to test small snippets of the program and fix bugs as soon as they were written. This saved us a substantial amount of time as it resulted in being able to find and fix bugs significantly faster.

3.2 Dataset

Our dataset was collected from different social media sites. The main data we worked with was Twitter data provided to us from our sponsor. This data set consists of many different factors. The first is the author id which is an identification number tied to a specific Facebook account. The next is the conversation id which is a number similar to the author ID but this is tied to the thread of a tweet rather than a user. The next part of the data is created at which is a timestamp that represents when the tweet was sent. The next piece of the dataset is public metrics which describes the interactions on the tweet. This shows how many likes, quote tweets, replies, and retweets there were on the given tweet. Next, is the variable name which is just the name associated with the twitter account. Next is the public metrics which looks into the aspects of the Twitter account, this stores the number of followers the user has, the number of accounts the user is following and listed count. Listed count refers to the number of public Twitter lists a user has been added to by other users. The final two aspects of the data are username which can also be referred to as a user's "@". And location is where the tweet sent from. This is not available on every tweet as users do not need to add this. This data was analyzed using pandas and NetworkX. We also created a Facebook plugin that can pull data from a Facebook page and creates a CSV file and then writes the Facebook page's data in to the CSV file that can be used to create cascades. The Facebook plug-in did create a challenge for the project, as in order to be able to pull data from Facebook Pages you need to have your Meta Developer Application reviewed. This process requires a Business Account which we did not have. We were able to work around this by testing the code on personal, empty Facebook accounts. This verified the success of the script. The end user of the application will need to have a business Facebook Account should they wish to pull data from Facebook.

3.3 Results and Analysis

- Our results surpass those of existing solutions as ours provides a slider that can be used to show the progression of the infection overtime. In other cascades, the visualization just shows the ending cascade. Our slider allows the user to show the progression of the infection event allowing the user to see who was infected at any point in time.

- The expected results can be seen in the figure above where as time progresses the Nodes in the cascade become more and more infected. The unexpected results were the time

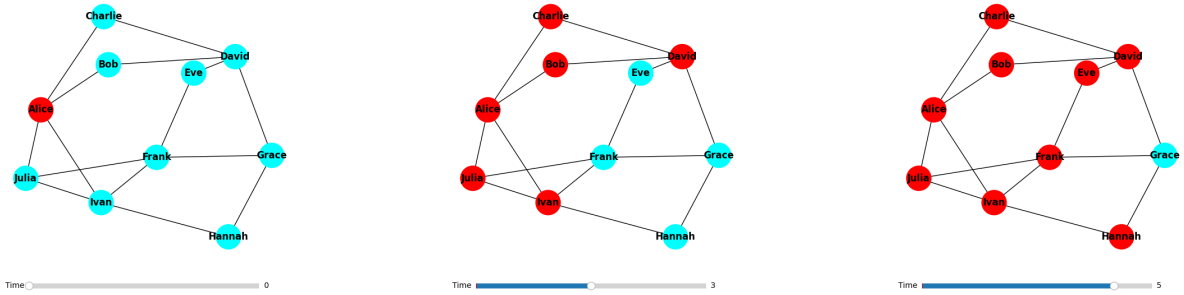

Figure 4: A row of three images with a single caption.

it took for nodes to become infected. The infection process began slow but the took of as the infection of nodes would be better described as exponential rather than a linear progression as we expected.

- Failure cases
  - Some of our projects limitations of our project is the time to create the cascades. When dealing with a dataset that is large (which our main testing data was) it took our program up to 10 seconds to compile and produce the cascade.
  - One of the biggest issues we found in our project testing was in using the Facebook Graph API without the proper permissions. The best way to overcome this issue is for the user of the application to use their business license to gain the permissions from Facebook.

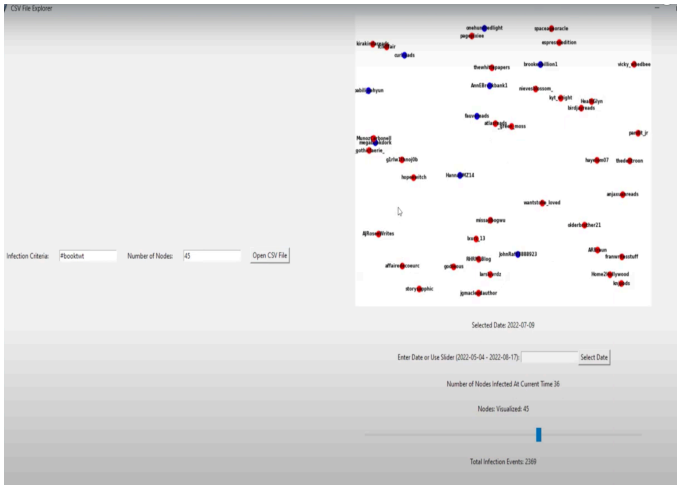

Figure 5: Cascade With Sample Data

4 Legal and Ethical Practices

4.1 Legal Considerations

- Data Privacy and Terms of Service: Verifying we are in compliance with the terms and service and privacy policies of Twitter and Facebook. Following privacy settings and data usage limits.
- Data Ownership: There needed to be a understanding of the ownership rights of the data when it pertains to user-generated content on Twitter and Facebook.

4.2 Ethical Considerations

- Not Using Third-Party Data Scrapers: There were existing web scrapers that can be used to pull Facebook Data. Many of these were not approved by Facebook itself and scraped the pages of Facebook without permission. It is for this reason we chose to use the Facebook Graph API to create out own application to pull data.
- Obtaining the proper permissions from Facebook to pull the data: This step was necessary as without permissions being granted any of the data we pulled from Facebook would have been not ethically sourced and would not be okay to use.
- One potential risk of the system is someone unethically using the system to analyze data. One thing that we have in place to prevent this is that in order to use our system's program to pull data from Facebook they need to have a Meta Developer Approved Application. In order for them to obtain this they need to be able to submit proof of a business they are working for and an explanation as to what they will be doing with the data.

5 Effort Sharing

When working on the project we decided that tasks should be divided depending on the strength and preference of the group member. Meetings were held weekly with Professor Magner where we would receive feedback and be given new tasks. After these check-in meetings we would meet to divide up the work that was assigned. Weekly in person coding sessions were also held in the library to demonstrate what each member was working on and assist each other in coding issues.

- Group Work: The main component of our group work was being an effective communicator and attending all group meetings. Meetings were held once per week on zoom with

Professor Magner to discuss progress, issues, and next steps of the project. In person coding sessions were also held weekly to show each other progress and help debug each others code.

- Alex Barry: Set up the GitHub environment. This was necessary as we needed a place for version control. Alex also was crucial in implementing the cascade. Alex also set up the python environment and initialized pytest. This was the main vehicle we used for testing the environment. Tested other group members code and assisted other group members with bugs and development issues.

- Blake Funderburke: The groups main data analyst. Responsible for writing a script that took the data given to us from Professor Magner and creating a graph with that data. This graph was essential to the project as it was leading point for creating a cascade throughout the project. This also demonstrated how infection events occurred. Developed a slider that can be used to adjust the time of the cascade at a certain time.

- John Syracuse: Was responsible for creating the Facebook plug-in that could pull data directly from Facebook pages and create CSV files that can be used to create other cascades. Created the Facebook Application needed in order to use the Facebook Graph API. Created the initial NetworkX Graph which served as a starting point for the final graphs. Created the Node class used in the cascade. Was mainly responsible for the final report compiling all of the information together.

1).

Table 1: Effort sharing

Team size	Joint efforts	Blake Funderburke	Alex Barry	John Syracuse
3	J (≈27%)	I (≈25%)	I (≈25%)	I (≈23%)

J = description of tasks jointly performed
I = description of tasks individually performed

6 Conclusion and Future Work

During the development of Data Gathering and a Modular Cascade Framework for Statistical Inference all group members were efficiently challenged and took significant work to accomplish. We believe that we addressed all of the things necessary for the project. Beginning with creating the cascade. We were able to create a cascade that is easy to read but also maintains the complexity of the problem and does not leave out any unnecessary information. We were also able to leverage time series data and be able to show the cascade and the infection data

at different timestamps. And finally, we were able to create an application that can pull data directly from Facebook with the user only needing their own Meta for Developers Account. The next steps of this project would likely be to make the cascade more in depth and could add more functionality and customization possibilities. Another possible thing that can be done in the future is to create a program that can pull data from Mastodon. The cascade can also be applied to academic research where an infection event rather than being a click of a hashtag, but a reference in a scholarly article.

7 References

[1] F. Xiaodong, Z. Qihang, and L. Yunkai, AECasN: An information cascade predictor by learning the structural representation of the whole cascade network with autoencoder, Expert Systems with Applications, vol. 191, Apr. 2022. (accessed February 19, 2024)

[2] What is data visualization? definition, examples, and learning resources, Tableau, https://www.tableau.com/learn/articles/data-visualization (accessed April 27, 2024).

[3] S. J. Haque, Graph theory 101, Science in the News, https://sitn.hms.harvard.edu/flash/2021/graph-theory-101/ (accessed April 27, 2024).

[4] What you need to know about network theory, USC Online Communication Degree, https://communicationmgmt.usc.edu/blog/what-you-need-to-know-about-network-theory (accessed April 27, 2024).

[5] Testing code: Types and benefits for software development, RSS, https://bluelight.co/blog/code-testing#:~:text=Code%20testing%20refers%20to%20running,or%20errors%20during%20the%20SDLC. (accessed April 27, 2024).

[6] Atlassian, What is version control: Atlassian Git Tutorial, Atlassian, https://www.atlassian.com/git/tutorials/what-is-version-control#:~:text=Version%20control%2C%20also%20known%20as,to%20source%20code%20over%20time. (accessed April 30, 2024).

[7] A. S. Gillis and S. Lewis, What is object-oriented programming (OOP)?, App Architecture, https://www.techtarget.com/searchapparchitecture/definition/object-oriented-programmi
~:text=Object%2Doriented%20programming%20(OOP)%20is%20a%20computer%20programming%20model,has%20unique%20attributes%20and%20behavior. (accessed April 30, 2024).

[8] Pytest: Helps you write better programs, pytest, https://docs.pytest.org/en/8.2.x/ (accessed April 30, 2024).

[9] M. Chen, H. Hauser, and P. Rheingans, Foundations of Data Visualization, http://link.springer.com/content/pdf/10.1007/978-3-642-28598-1.pdf (accessed May 2, 2024).

[10] D. Unzueta, Data Visualization theory: An introduction, Medium, https://towardsdatascience.com/data-visualization-theory-an-introduction-a077c0d80498 (accessed May 2, 2024).

[11] D. C. Sordillo, Cascade Networks, generalized neural networks, and approximation of functions, Vanderbilt University Institutional Repository, https://ir.vanderbilt.edu/handle/1803/17766 (accessed May 2, 2024).