Learning the Popularity Prediction in Information Cascades

Final Report

Team: S23-35
Client and Advisor: Dr. Goce Trajcevski

Ian Johnson - Frontend
Will Postler - Client Interaction
Paul Brinkmann - Database
Evan Gossling - Backend
Bailey Gorlewski - CI/CD, Mapbox

sdmay23-35@iastate.edu
https://sdmay23-35.sd.ece.iastate.edu/

Revised: 4/29/2023
Version: 2

Executive Summary

Development Standards & Practices Used

- Scrum Methodology
- IEEE 830 - Software Requirements Specifications
- IEEE 1016 - Software Design Descriptions
- IEEE 12207 - Software Life Cycle Processes
- IEEE 1028 - Software Reviews and Audits

Summary of Requirements

Constraints:

- Data parsed under 15 seconds and visualizing outputs of models for graphs made under 15 seconds
- A server to be running with minimum downtime (99.9 % uptime) to host the website and database
- Website must have decent performance (takes less than a minute to load a page)
- Graphs are able to zoom in and out with at most a 5 second delay

Functional/Specification:

- Display graphs/charts representing cascade data
- Display geo-locations of cascade graph data accurately
- Analize / extract from geographical data from author's affiliations
- A website to allow data sets to be inputted and graphs to be outputted

Resource/Physical:

- Visualization tools/frameworks

Aesthetic:

- Website must be designed in a pleasing manner so the webpage is nice to look at and easy to read/understand

User Experiential:

- Website must be easy to use and navigate
- Multiple users need access to the backend

Economic/Market:

- Application and provided services function well on the server provided to us

Environment:

- Application makes efficient use of power on the server and each webpage

Applicable Courses from Iowa State University Curriculum

- COM S 227: Object-Oriented Programming
- COM S 228: Introduction to Data Structures
- COM S 309: Software Development Practices
- COM S 311: Introduction to the Design and Analysis of Algorithms
- SE 329: Software Project Management

New Skills/Knowledge acquired that was not taught in courses

- Mapbox API
- Flask Framework
- Information Cascades
- Machine learning development

Table of Contents

Dictionary

Information Cascades: When a person makes a decision made solely on the decisions of other people, while ignoring their own personal knowledge to the contrary.

Cascade Graph: A graph that shows you how an initial value is impacted by intermediate values — either positive or negative — and results in a final value. In the context of the project these cascading graphs are generated based on predictive models.

Machine Learning: The use and development of computer systems that are able to learn and adapt without following explicit instructions, by using algorithms and statistical models to analyze and draw conclusions from patterns in data.

Figures and Tables

1 Team

1.1 Team Members

Ian Johnson

Will Postler

Paul Brinkmann

Evan Gossling

Bailey Gorlewski

1.2 Required Skill Sets for Your Project

- Ability to use a Framework with Python to build our website on
- Able to effectively communicate with team members
- Server/database management capabilities with MYSQL
- Ability to make graphs and charts from data and able to integrate them into the website
- Ability to integrate Mapbox into our website
- Knowledge of frontend tools (i.e. HTML, JS, and CSS)
- Ability to integrate Machine Learning algorithms for Cascading Prediction Models into our website

1.3 Skill Sets covered by the Team

Ian: Communication, MYSQL, HTML

Will: Communication, MYSQL, Frontend tools, Graphs/Charts

Evan: Communication, Framework tools, Graphs/Charts, Frontend, Python

Paul: Communication, MYSQL, Frontend tools

Bailey: Communication, MYSQL, HTML, Python, Graphs/Charts

1.4 Project Management Style Adopted by the team

Our team utilized the Scrum/Agile Methodology, this is because the scrum methodology has regular team meetings and prioritizes communication. Our team worked on components that are highly coupled together and required frequent and effective team communication. The scrum methodology is also more robust and allows teammates to communicate issues and setbacks encountered during development.

1.5 Project Management Roles

Ian Johnson - Frontend

Will Postler - Client Interaction

Paul Brinkmann - Database

Evan Gossling - Backend

Bailey Gorlewski - CI/CD, Mapbox

2 Introduction

2.1 Problem Statement

In the current digital age the Internet, social platforms such as Twitter, Weibo, WeChat, and more have become the main source of information in people's daily lives. Various news, events, and posts are spread through social networks. Similarly, information is spread around the academic world through papers. These papers get read, shared, and cited, similarly to how tweets get liked, retweeted, and quoted. Therefore, predicting the effect of an individual paper after a certain time period (in terms of some of the metrics previously mentioned) has garnered the attention of academics and publishers. Measuring and predicting the propagation of papers shows how much of an impact each one has had / may have, which is good for both evaluating a paper's quality and relevancy (e.g. if it's 20 years old and never been cited then it's likely irrelevant or poorly executed).

For example, the American Physical Society (APS) contains scientific papers published by APS journals (https://journals.aps.org/datasets). Every paper in the APS dataset and its citations form a citing cascade. Many researchers work on addressing the problem of information cascades using this APS data set. However, in the past, developed models (e.g. deep learning techniques) have not payed attention to:

1. The users' need to look up data in the APS or similar data set
2. Visualizing citation data on a map
3. Handling updates to the dataset and informing users

The objective of this project is to develop an end-to-end system that will provide such functionalities for the users interested in visualizing locations related to popularity cascades in the context of scientific paper citations.

2.2 Intended Users and Uses

There are a few broad categories of users that would benefit from such systems:

1. Individual scientists can see what papers are likely to gain citations, or look for prolific papers and predictions for related papers
    a. See which papers have large amounts of citations.
    b. See visualizations of data across the country based on these citations
    c. View trends over time of citation usage across country
    d. See how their own work stands against other researchers.
    e. Look for more prolific research topics.
2. Funding institutions can see which universities are likely to gain recognition due to the increase of citations and reasons about their geographical distributions
    a. See which papers have large amounts of citations.
    b. See visualizations of data across the country based on these citations
    c. View trends over time of citation usage across country
3. University administrations that want to compare scientific impact of its faculty personal in with peer institutions based on geographic whereabouts
    a. See which papers have large amounts of citations.
    b. See visualizations of data across the country based on these citations

c. View trends over time of citation usage across country
d. Provide funding based on data

Personas:

Individual Scientist - Sam

Sam is a PhD researcher working towards tenure at their University. Sam wants to make sure that they are researching topics and publishing papers that will be cited frequently to increase their notoriety within their field of academia.

Funding Institution - Frankie

Frankie is a staff member at an institution that provides funding for research projects. Frankie needs to be able to see papers with large amounts of citations, to see which topics their institutions can provide funding for in the future. This will allow Frankie to successfully fund research that will have a larger impact on academia/overall world research.

University Administrator - Alex

Alex is an administrator who enjoys promoting their institution. Alex likes to back up their praise with details about how their institution has written research papers that have been cited across the country/world. This data could allow Alex to recruit and retain researchers at their institution.

2.3 Requirements & Constraints

Functional/Specification:

- Display graphs/charts representing cascade data
- Display geo-locations of cascade graph data accurately
- Analize / extract from geographical data from author's affiliations
- A website to allow data sets to be inputted and graphs to be outputted
- Data parsed under 15 seconds (constraint) and visualizing outputs of models for graphs made under 15 seconds (constraint)

Resource/Physical:

- A server to be running with minimum downtime (99.9 % uptime (constraint)) to host the website and database
- Visualization tools/frameworks

Aesthetic:

- Website must be designed in a pleasing manner so the webpage is nice to look at and easy to read/understand

User Experiential:

- Website must be easy to use and navigate
- Multiple users need access to the backend
- Website must have decent performance (takes less than a minute to load a page - constraint)
- Graphs are able to zoom in and out with at most a 5 second delay (constraint)

Economic/Market:

- Application and provided services function well on the server provided to us

Environment:

- Application makes efficient use of power on the server and each webpage

2.4 Engineering Standards

IEEE 830 - Software Requirements Specifications:

Justification: It is essential that we have set software requirement specifications, in order to be clear in the scope of our project, and to have set milestones for what was accomplished.

IEEE 1016 - Software Design Descriptions:

Justification: We fulfilled this standard by having a concrete set of design documents, it was important that our full design process was documented.

IEEE 12207 - Software life cycle processes:

Justification: For this project we utilized an agile software development life cycle that allowed us to continuously improve/test our software. Having a set SDLC process allowed us to be clear in how we went forward in our development.

IEEE 1028 - Software Reviews and Audits:

Justification: Reviewing the code written by each team member allowed us to catch bugs that may not be otherwise found, and provided a 2nd set of eyes on what has been contributed. Code reviews are an essential part of the SDLC, and are used in most enterprise software projects.

3 Project Plan

3.1 Project Management/Tracking Procedures

Our project adopted an agile methodology for our project management. The main goal of this project was to evaluate a kernel-based structural-temporal cascade learning model to explicitly estimate and encode the structural similarity of cascades with graph kernels. To accomplish this we decided it would be best to use the agile methodology because we needed to constantly test and update our software while developing it. We also had cross functional teams that were working on multiple segments at the same time. Since these segments are closely related, agile was preferred as it allowed the needs for each segment to constantly develop.

We used GitLab to track our team's progress. This allowed each member to stay up-to-date on the current project status and completed tasks. We also used Discord to communicate, as it was a convenient way to easily communicate online. Discord also allowed us to separate several project areas such as frontend and backend channels so members can easily differentiate project information and meeting details.

3.2 Task Decomposition

Task	Title	Description
0	Design Infrastructure	Formalized the frameworks and application structures.
1	Create Local Environment	A local environment implementing the structure from task 0 is created and shared among developers.
2	Initialize Database	An SQL database is initialized with given data.
3	Implement Algorithms	Make sure algorithms are implemented and produce values that are expected based on algorithms given to us.
4	Design UI	Design a web frontend that allows users to customize their queries.
5	Implement UI Design	Implement and develop the UI as designed in task 4.
6	Create Queries	Have all necessary queries for each function finalized.
7	Implement backend ML logic	All algorithms for parsing and representing data are implemented in the backend.
8	Add Backend REST Endpoints	Endpoints to access data from task 7 from the frontend are available.
9	Add UI functionality	Add different assorted quality of life functionality to the UI.

10	Create Production Environment *Data Visualization	Add data visualization tools to the frontend.
11	Help documentation	Include documentation to help users understand what they are looking at and troubleshoot problems they run into using the application.
12	Presentation	Give a presentation about the project.

Table 3.1 Task Decomposition

3.3 Project Proposed Milestones, Metrics, and Evaluation Criteria

- Milestone 1: Architecture Design (Oct. 28)
    - We had a diagram that shows how the frontend and backend will work together. Our diagram will be complete, but also flexible to account for any changes that need to be made in the future.
- Milestone 2: Finalize Algorithm Solutions (Nov. 11)
    - By this date, we choose algorithms we will use to determine how to generate the population cascades for the dataset.
- Milestone 3: Finalize Design Document (Nov. 18)
    - The design document was comprehensive and contained all information necessary for the development of the software. The document contained all design plans for the software, how the team will work together to develop the software, and the plans for testing the quality of the software.
- Milestone 4: Revising Design Selections and Criteria (Dec. 2)
    - We revised and finalized the requirements and criteria for success to be as accurate as possible before we ventured into the second semester.
- Milestone 5: Complete Unit Testing (Dec. 9)
    - We implemented test cases for each component of the application that was used to ensure proper output during development. Each component was tested for vulnerabilities, things that will cause the server to crash, and general correctness.
- Milestone 6: Finalize Presentation (Dec. 9)
    - We ensured our presentation for the faculty judges has accurate information and enough information. We have gone through a few practice runs to ensure each member knows what they will be saying and which ideas they will be communicating.

- Milestone 7: Complete Integration Testing (Apr. 25)
    - We tested each component for compatibility and vulnerabilities. We also tested component interactions for any vulnerabilities or problems that could be caused.
- Milestone 8: Alpha Release (Apr. 25)
    - A simple web application that displays the information from at least one dataset ran through at least one learnt algorithm. Show the map and provided graphs, and have decent descriptions for each graph.
- Milestone 9: Final Software Version (Apr. 29)
    - The final version was finished. UI was finalized, bugs ironed out, functionality finalized, and design requirements met.
- Milestone 10: Final Presentation (May 3)
    - The final version of the presentation included a demonstration of the completed project highlighting different features of the software.

3.4 Project Timeline/Schedule

CPRE 491	10/14	10/21	10/28	11/4	11/11	11/18	11/25	12/2	12/9			
Task	Week 8	Week 9	Week 10	Week 11	Week 12	Week 13	Week 14	Week 15	Week 16		Color	Area
Design Infrastructure												Frontend
Design UI												Backend
Create Local Environment												Both
Presentation												

CPRE 492	1/20	1/27	2/3	2/10	2/17	2/24	3/3	3/10	3/17	3/24	3/31	4/7	4/14	4/21	4/28	5/5		
Task	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9	Week 10	Week 11	Week 12	Week 13	Week 14	Week 15	Week 16	Color	Area
Initialize Database																		Frontend
Implement Algorithms																		Backend
Implement UI Design																		Both
Create Queries																		
Implement backend ML logic																		
Add Backend REST Endpoints																		
Add UI functionality																		
Data Visualization																		
Help documentation																		
Presentation																		

Table 3.2 Project Gantt Chart

3.5 Risks And Risk Management/Mitigation

For each of our tasks, the risk probabilities were pretty low. We had experience with most of these tasks and therefore knew how to implement them.

Task	Title	% Risk	Reason	Mitigation Strategy
0	Design Infrastructure	0.1	Planning Stage	N/A
1	Create Local Environment	0.2	Initialization Stage	N/A
2	Initialize Database	0.4	Initialization Stage	N/A
3	Implement Algorithms	0.5	Initialization Stage	Made sure algorithms were implemented and produced values that are expected based on algorithms given to us.
4	Design UI	0.3	Planning Stage	N/A
5	Implement UI Design	0.4	Initialization Stage	N/A
6	Create QueriesQueries	0.7	Performance Risk	Made sure that queries met performance levels. Considering that there is a large amount of data, timely queries were very important.
7	Implement backend ML logic	0.9	Performance Risk	Algorithms needed to be thoroughly tested to make sure they meet performance criteria.
8	Add Backend REST endpointsEndpoints	0.5	Tool Failure Risk	Issues with REST tools may arise. Other tools may need to be looked into for backup options.
9	Add UI functionality	0.4	Performance Risk	N/A
10	Create Production Environment *Data Visualization	0.5	Performance Risk	Possible issues with performance. Mitigation may be different tools, increased testing, and possibly no implementation.
11	Help documentation	0.1	Documentation	N/A
12	Presentation	0.1	Documentation	N/A

Table 3.3 Risks and Risk Management

3.6 Personnel Effort Requirements

The following Table 3.4 outlines the total person-hours used to complete each task in the project.

Task	Title	Hours	Explanation
1	Design Infrastructure	90	The frameworks and application structure form the foundation on which the application is built.
2	Create Local Environment	6	A local environment was necessary to develop in and verify proper functionally of the application before deployment.
3	Initialize Database	12	SQL databases with the given data were initialized with data from the given datasets. Had to reformat data to a uniform format.
4	Implement Algorithms	30	We took existing algorithms, figured out how they work, were able to programmatically and flexibly send data to them, and analyzed their output and turned that into data we can display.
5	Design UI	60	Created visual design for the frontend UI.
6	Implement UI Design	40	The UI forms a large portion of this project. This is the medium in which the users of the app interact with the data.
7	Create Queries	10	The queries used with the SQL database will be somewhat complex to account for user customization.
8	Implement backend ML logic	15	The backend needed to take the algorithms implemented above, and utilize them based on the data inputted.
9	Add Backend REST Endpoints	6	This simple task made accessing the data from the frontend much easier.
10	Add UI functionality	15	Fully functional ability of frontend to communicate with backend and data requests.
11	Data Visualization	20	The app has graphs and maps to reflect the user's queried data.
12	Help documentation	15	Documentation needed to be created to comment code, as well as explain how to use our web application.
13	Presentation	10	A final presentation needed to be created in order to showcase the functionalities of our project, as well as the design process.

Table 3.4 Personnel Effort

3.7 Other Resource Requirements

For additional resources, we needed a server to manage the machine learning algorithms and a server to manage the website and database. Besides that, there are no other resources that were needed besides the open source software we utilized (APIs like the Mapbox API).

4 Design

4.1 Design Context

4.1.1 Broader Context

One of the main objectives of information cascade popularity prediction is to forecast the future size of a cascade given the observed propagation information. It is an enabling step for many practical applications (e.g., advertisement, academic writing, etc.). Recent advances in neural networks have spurred a few deep learning-based cascade models, which preserve the structural features of information cascades with node embedding and graph neural networks. However, most of the efforts in cascade graph learning as well as its internal temporal dependency, mainly focus on node-level similarity learning, ignoring the structural equivalence among different sub-graphs that are more informative for information diffusion prediction. The main goal of this project was to evaluate a kernel-based structural-temporal cascade learning model to explicitly estimate and encode the structural similarity of cascades with the graph kernels. We also employed a non sequential process to address the temporal dependency which, in turn, can be used to facilitate information popularity prediction. The methodology was evaluated on real datasets, for which a (front-end) visualization was one of the objectives of this project.

Area	Description	Examples
Public health, safety, and welfare	How does your project affect the general well-being of various stakeholder groups? These groups may be direct users or may be indirectly affected (e.g., solution is implemented in their communities)	Increasing/reducing exposure to pollutants and other harmful substances, increasing/reducing safety risks, increasing/reducing job opportunities
Global, cultural, and social	How well does your project reflect the values, practices, and aims of the cultural groups it affects? Groups may include but are not limited to specific communities, nations, professions, workplaces, and ethnic cultures.	Development or operation of the solution would violate a profession's code of ethics, implementation of the solution would require an undesired change in community practices
Environmental	What environmental impact might your project have? This can include indirect effects, such as deforestation or unsustainable practices related to materials manufacture or procurement.	Increasing/decreasing energy usage from nonrenewable sources, increasing/decreasing usage/production of non-recyclable materials
Economic	What economic impact might your project have? This can include the financial viability of your product within your team or company, cost to consumers, or broader economic	Product needed to remain affordable for target users, product creates or diminishes opportunities for economic advancement, high development

	effects on communities, markets, nations, and other groups.	cost creates risk for organization

Table 4.1 Broader Context Considerations

4.1.2 Prior Work/Solutions

To our knowledge, there was no visualization tool that can provide predicted mappings of the popularity of scientific works. Some agencies like NSF have visual analytics but only for existing projects, not for predicted ones. While there exists certain works addressing prediction models we will focus on relatively recent results that have developed ML models for predicting the scientific impacts. Our two main sources for the algorithms and data are from these two papers, which detail using algorithms based on Recurrent Cascades Convolutional Networks (CasCN) (Information Diffusion Prediction via Recurrent Cascades Convolution) and Variational Cascades (VaCas) (Variational Information Diffusion for Probabilistic Cascades Prediction) to predict cascades.

- X. Chen, F. Zhou, K. Zhang, G. Trajcevski, T. Zhong and F. Zhang, "Information Diffusion Prediction via Recurrent Cascades Convolution," 2019 IEEE 35th International Conference on Data Engineering (ICDE), 2019, pp. 770-781, doi: 10.1109/ICDE.2019.00074.
    - https://ieeexplore.ieee.org/document/8731564
- F. Zhou, X. Xu, K. Zhang, G. Trajcevski and T. Zhong, "Variational Information Diffusion for Probabilistic Cascades Prediction," IEEE INFOCOM 2020 - IEEE Conference on Computer Communications, 2020, pp. 1618-1627, doi: 10.1109/INFOCOM41043.2020.9155349.
    - https://ieeexplore.ieee.org/document/9155349

4.1.3 Technical Complexity

The architecture of the system provided a frontend (UI), with a backend that does most of the processing, data storage, etc. In addition, different components had distinct subsystems that were integrated. For example, the UI had to combine map display, with the various menu options. The design also had to consider the execution of multiple algorithmic solutions, as well as integrating the heterogeneous data. The UI also utilized a mapping API Mapbox to show the actual map data.

The challenges of this project involved coupling of outputs of cascade graph prediction algorithms with displaying geospatial values of attributes that are secondary to the ML solution. This allowed users to see geospatial visualizations of the prediction algorithms, and make decisions based on them.

4.2 Design Exploration

4.2.1 Design Decisions

It has been noted that Google Maps has issues with its API. It has more limitations (with the free version) and also has less performance than MapBox. It is also less intuitive to use. Therefore, we used Map Box as it is more intuitive, especially for preliminary projects.

An important design decision we chose is the data storage management we used, straightforward relational (e.g., mySQL) to publicly available spatial databases (e.g., QGIS). We decided on mySQL as it was the most familiar to our team.

We also decided on which middle-ware technologies we needed to select for effective generation of objects between frontend and backend (JSON, XML etc.). We decided to use JSON as it was more compatible with our use of Mapbox.

4.2.2 Ideation

To generate ideas for our design decisions we created a Lotus Blossom.

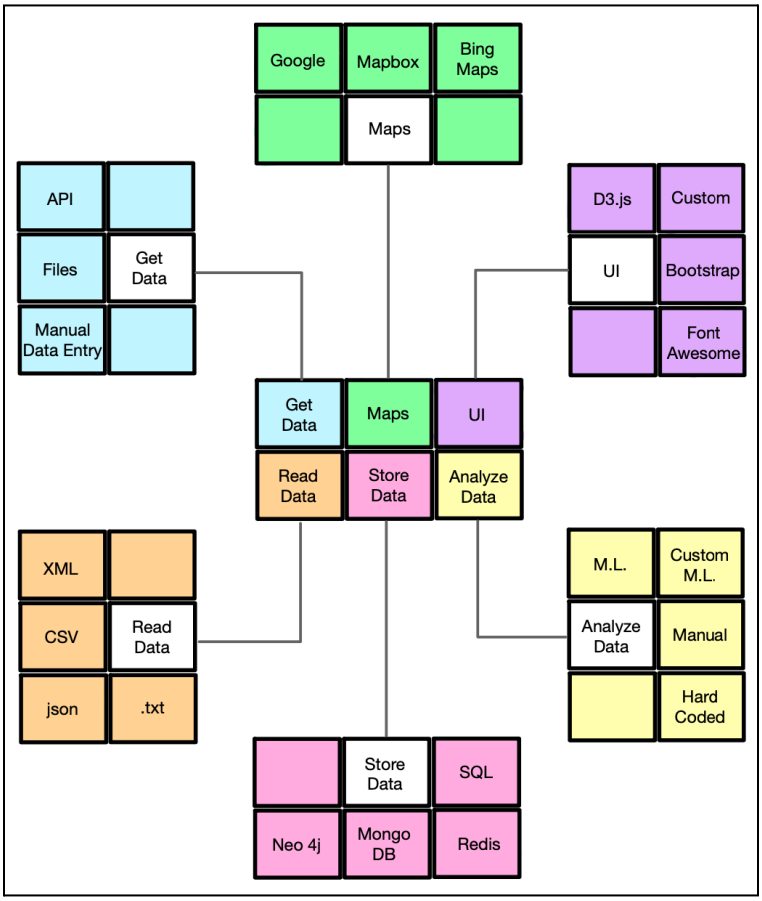

Figure 4.1 Ideation Lotus Blossom

For mapping, we had a few options to consider. One of the first choices we eliminated was Fencer, an API that we had heard of, but was discontinued in 2018. That limited us to one of three choices: Mapbox, Google maps, and Bing maps. We looked at a variety of factors on each of these different APIs (decision matrix below), and ended up picking Mapbox.

For our “Get Data”, the design choices were pretty simple. It does not make sense to choose “manual data entry” for our website, so we can immediately eliminate that. Files and API are both good choices. Using an API was selected as it was a good option for when we want to pass the backend data to the frontend for our charts, maps, and graphs.

For our “Store Data” part of the Lotus Blossom, we considered using a SQL database, Redis, MongoDB, or Neo4j. Each of these database options provide their own strengths and weaknesses, so integration testing will be needed to see which one will perform the best. We decided on SQL as it was most familiar to our team.

4.2.3 Decision-Making and Trade-Off

After our ideation process to come up with multiple map API options, we used the following decision matrix to finalize our decision (Table 4.2).

	Selection Criteria	Weight	Google Maps	Mapbox	Bing Maps
1	Documentation	0.4	9	9	6
2	Ease of use	0.3	7	7	6
3	Performance	0.2	8	9	6
4	Afordability	0.05	4	7	10
5	Aesthetic	0.05	7	8	3
6	Total	1	35	40	31

Table 4.2 Map API Decision Matrix

As a team we created several considerations and criteria for the map API we would use including available documentation and performance. Then we researched each map API on each criterion. We then chose Mapbox for our project, because it had the highest weighted score in our decision matrix (Table 4.2). Mapbox has similar documentation and tutorials to Google Maps, however, Mapbox has a better performance and price structure for our project. Overall, Mapbox will be best suited for our project’s needs.

4.3 Final Design

4.3.1 Overview

Our design is, in the most general sense, a website. It consists of the following pages: a homepage, an about us page, a query creation page, and a visualization page. The homepage contains a brief tutorial / explanation of how to use our tool and the purpose of our project, and the about us page gives information about our team. The main functionality we are providing, as described in later sections, is provided by the remaining two pages. Each of them works towards providing the core functionality we are trying to achieve, and allow for things like the creation of Machine Learning queries and displaying of predictions (location and otherwise) of information cascades.

Current version for the design is targeting the main components of a system that will enable the desired behavior. To better illustrate the subsequent sections of this document we provide a use case illustration below:

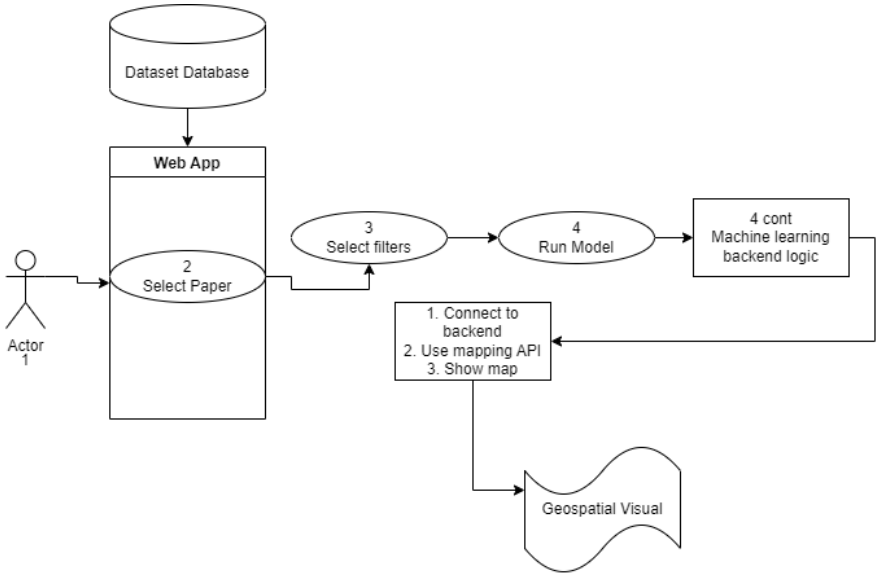

Figure 4.2 Use Case Diagram

4.3.2 Detailed Design and Visual(s)

Based on our use case model (cf. Figure 4.2), here is our diagram of our system and subsystems:

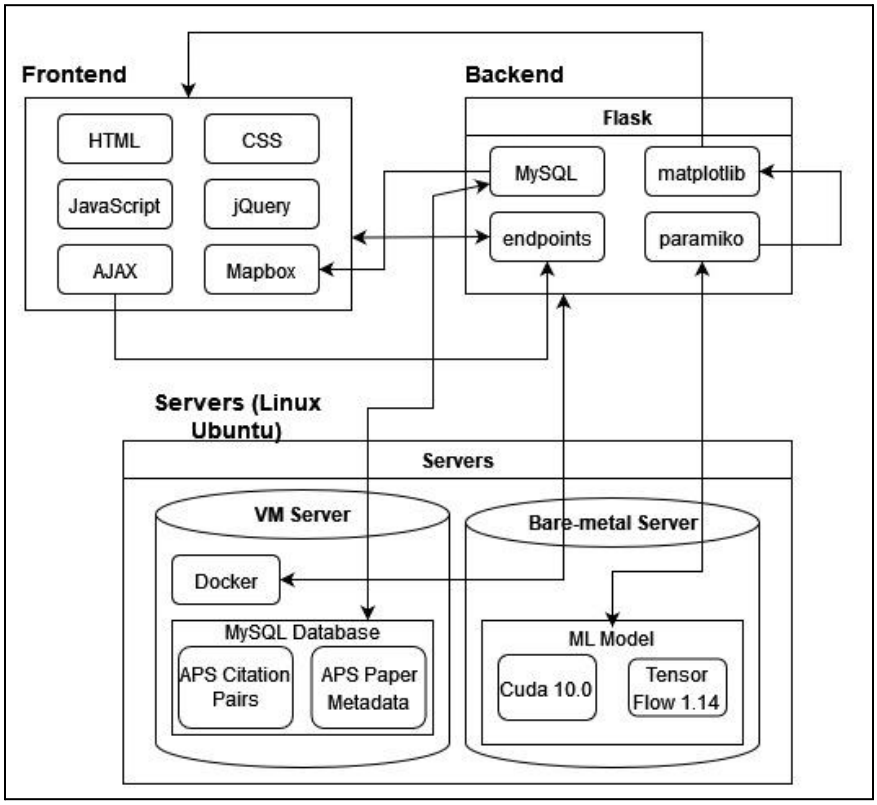

Figure 4.3 System Block Diagram

Now we describe each subsystem and their interactions:

4.3.2.1 Frontend

HTML:

This subsystem is used in the frontend and is responsible for displaying our documents. It is the basis of our web application and everything is displayed using our HTML pages or built into the HTML pages themselves. The HTML pages are rendered using Flask via our set endpoints, and any necessary data is passed to the HTML page using Flask. The other frontend subsystems: CSS, JavaScript, jQuery, Ajax, and Mapbox are all integrated or displayed using HTML.

JavaScript:

This subsystem is a programming language usable with HTML. It allows us to bring scripting functionalities and an object-oriented language into our HTML pages. It utilizes many third-party libraries such as jQuery, Ajax, and Mapbox.

CSS:

This subsystem is used in the frontend and is responsible for styling our web applications. It turns what would be a harsh, basic, and inconvenient User Interface into a user-friendly, eye-appealing, and easy-to-use User Interface.

jQuery:

This subsystem is used in the frontend as a library of JavaScript. It is used for event handling such as select field changes or button on clicks. It also handles some CSS animations and Ajax.

Ajax:

This subsystem is used in the frontend and is another library of JavaScript. It is used to send POST requests to utilize some of our endpoints in Flask.

Mapbox:

This subsystem is used in the frontend and is our last JavaScript library. Mapbox is used to display pins on a map of the world. We use this library to pinpoint locations that a selected paper has affiliations with.

4.3.2.2 Backend

Flask:

This subsystem is used as the backbone and foundation of our web application and backend. We use this to add more sophisticated coding into our web app as it allows us to utilize all of Python's libraries. Because of this, we are specifically able to use MySQL, matplotlib, and paramiko. We also use Flask to run our web app and route our backend endpoints. Our Flask application is containerized within Docker and then deployed on one of our VMs to host our web application.

MySQL:

This subsystem is a python library used within Flask, which is our framework and backend. We use the MySQL library to connect to our MySQL database within one of our VMs. This way, we are able to retrieve data from our database which is necessary for obtaining papers from the APS dataset and metadata of a selected paper.

Matplotlib:

This subsystem is a Python library used within our Flask framework. We use matplotlib to generate our graph where the line of interest is plotted using the variables obtained from our ML model and the equations from which our project is based on.

Endpoints:

This subsystem is used in the backend as a part of Flask. It is responsible for setting the endpoints used within our web application. These endpoints are set up into two categories, returning and visualizing a HTML page, sometimes with some python logic beforehand to pass data through to the HTML page. This generally involves using other libraries such as MySQL, paramiko, and matplotlib. The second type of endpoints are used purely for Python logic to supplement our web app.

Paramiko:

This subsystem is used in the backend as a part of Flask. It is a Python library that allows us to communicate with our VM servers. By using this we were able to SSH into our server and execute commands on it. We specifically use paramiko to execute our ML script to obtain the necessary values that allow us to generate the predictive graph with the expected number of citations.

4.3.2.3 Virtual Machines

Docker:

This subsystem is used in our VMs, it is responsible for containerizing our web application and hosting it on our server. We Dockerize our Flask application and then run the Docker image on our server. This then allows our web application to be hosted on the ISU network and accessible to anyone on the network.

MySQL Database:

This subsystem is used in our VMs and is responsible for storing our two APS datasets. The first dataset is used to store all of the APS papers, and the second is used to store the metadata of the respective papers. We utilize both datasets to plot the locations affiliated with a specific paper using Mapbox.

Machine Learning Models:

This subsystem is used in the backend and is stored in our physical VM. It is responsible for using the APS data to train a model which will then be used to generate the “expected number of citations over time” graph using our input parameters.

4.3.3 Functionality

This application is intended to be used in order to generate citation predictions and geospatial visualizations. Users first select the paper they intend to generate predictions for, from the APS dataset. The user will then select the number of predictions that they would like to make, and the timespan for those predictions. After the user launches the query, a geospatial visualization, and a graph will be generated to show the expected number of citations, as well as the locations of cited institutions. The figure below shows an example of a page generated by a query.

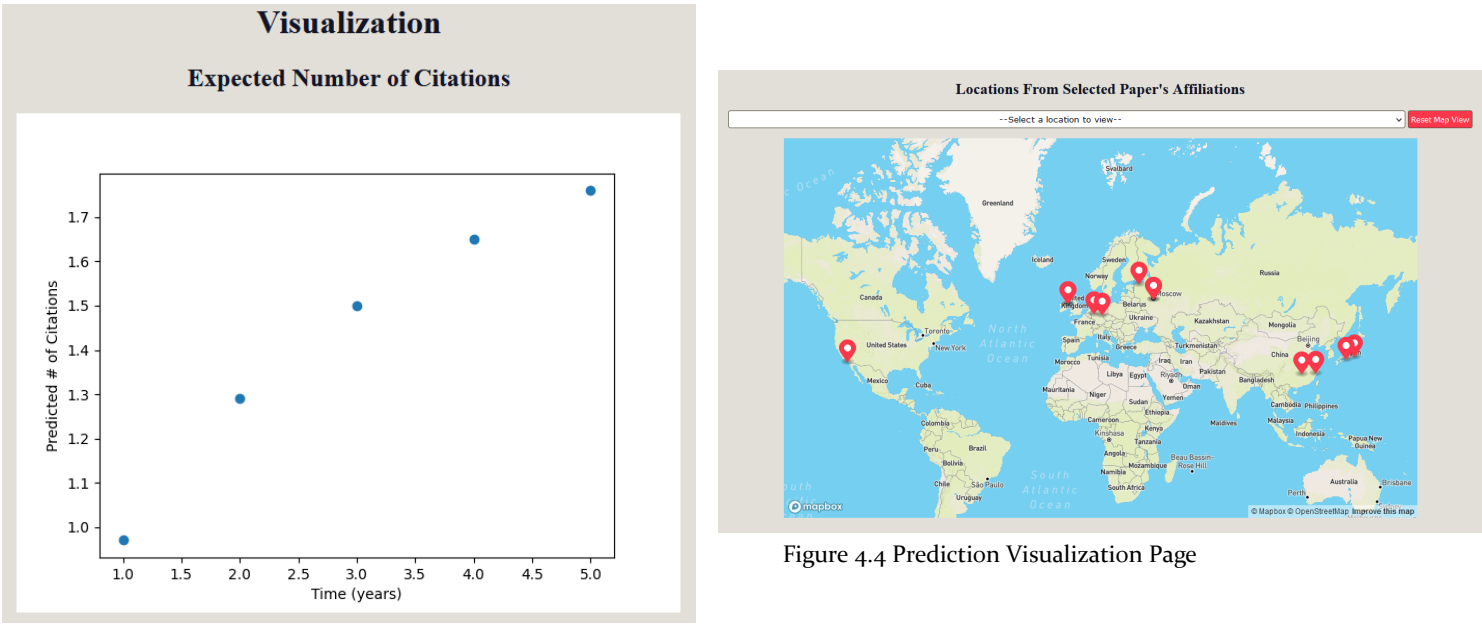

Figure 4.4 Prediction Visualization Page

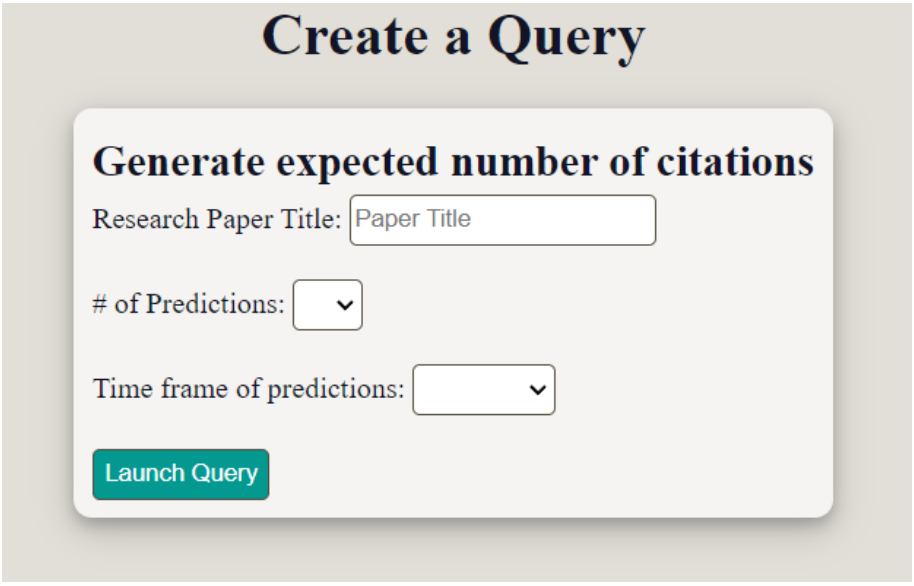

Figure 4.5 Query Creation Menu

The above figure shows the paper search functionality, as well as the filter drop downs for the query generation functionality. The search feature utilizes an autocomplete functionality in order to allow users to precisely select the paper they intend to generate predictions for. This feature also makes sure that users are not selecting papers that are not included in the dataset.

4.3.4 Areas of Concern and Development

We had constant discussions with our client to ensure that all our requirements meet their needs. For example, one of the biggest needs expressed was to be able to visualize geo-location data of citations on a map. We detailed that decision in Section 4.2.3 of this document. Just like with that decision, we went through the creation of evaluation criteria for each decision, allowing us to act confidently in our decisions. We also went over our decisions and why we made them with our client to ensure they felt that their needs were met.

While we considered the different frameworks and technologies to use (cf. Section 4.4), notwithstanding their benefits for the implementation, it turned out that we needed to gain more experience and hands-on practice with some of them. Our primary concern was becoming proficient with them and integrating the sub-components of the overall systems. We continued to communicate with our client in order to make sure that our product design meets their requirements.

The plan was to allocate tasks corresponding to getting familiarized with technologies, as well as portions of time among the team members who will be in charge of their corresponding role in the sub-components. We also had discussions related to integration testing scenarios.

4.4 Technology Considerations

MySQL: Most of the team already knew how to use it and it is pretty standard within the industry. We needed to store data and this was the best way to do it.

Mapbox: Mapbox had similar documentation and tutorials to Google Maps (competing software), however, Mapbox had a better performance and price structure for our project.

Flask: This was a very popular framework and easy to set up and use. There were also an abundance of tutorials online for its use, so we can easily learn how to use and implement it. Flask was already known to a team member which created a more fluid implementation.

4.5 Design Analysis

For the design, we decided on the libraries and languages we used in the project. This includes the frameworks such as Flask that were responsible for the website endpoints. We also planned out how all the smaller components interact including getting the model data from our Bare-metal server to our VM 1 website server using paramiko. We also planned out an agile schedule that allowed for these components to be developed simultaneously with feedback from our client.

4.5.1 Security Concerns and Countermeasures

While we recognize the importance of cybersecurity and its concerns, this project is mostly to act as a proof of concept, and does not have any large scale security implications. One possible physical security concern is that our model requires a high-end GPU in order to function properly, so anyone

taking on this project would need to make sure to keep that properly secured. In regards to cybersecurity, the only real concern that we faced while implementing our design, was that our research paper search functionality was originally vulnerable to SQL injection attacks, as the terms searched by the users were run as direct SQL queries. This concern was mitigated by implementing a system to escape all user input, to make sure users could not run their own queries.

5 Testing

In this section we present the details of our test methodology that was used throughout the development of the project. We note that the functional and nonfunctional requirements were described in Section 1.1 and they were translated into respective system components in Section 4.3. We will refer to Figure 4.2 and Figure 4.3 of Section 4 when describing the envisioned testings.

We note that cost related issues are not addressed since the objective of this project is to provide a proof of concept implementation, not a full fledged product. We also note that while following the good practices for testing, our plan was adapted to the specific needs of the project. In that regard, it is comprehensive with respect to the requirements provided by the client.

5.1 Unit Testing

The first group of unit testing corresponds to different components of the UI. These components include: the homepage, Prediction Configuration, and User Visualization systems. The second group of unit testing corresponds to the backend components. These components include: the Location Generation System, Prediction Management System, and Data Management System.

For each of these systems we checked whether the input is appropriate and the accuracy of the component output. For example, the User Visualization component was tested on if the component can accurately display data to the user and can be interacted with.

Since different team members worked on different units, and moreover different stages and their different milestones, certain units were tested early (eg. displaying maps). Each group member described (and, where applicable, implemented) unit tests as each part was started.

The unit testing tools depend on the software utilized in each component. For example, most of our application was written in Flask, so we used Pytest to do most of our unit testing.

5.2 Interface Testing

We have many interfaces in our design. The biggest one we had to test was between the front-end and back-end. The most important integration path between those two is the integration between the front-end Mapbox map and the back-end ML model. This path traverses from our ML model located on our bare-metal server to our Flask endpoints using paramiko, then to our front-end Mapbox map. We were able to test each component in the chain. For example, we ensured that the user imputed prediction queries were able to pass off the correct configuration input that was required for the predictive ML model. Testing to ensure that the configuration entered matches the configuration being received was very important.

5.3 Integration Testing

Unit testing and Interface testing are both critical and were done before we moved onto Integration testing. There are several critical paths, forward path (front UI to backend) and backward path (backend to UI). We proceeded with full tests in terms of appropriately identifying user requests, and correctly checking the answers and the display/locations. Most integration tests were completed during the development stage, ensuring that our application running in a docker container was able to interface with a MySQL database on the raw server and a physical machine that was running the model.

5.4 System Testing

We drafted scenarios that included all of the testings described in the previous Sections (5.1 - 5.4) in a seamless scenario. For example, we have a full test that involves a user selecting a paper from the dataset, which then we generate the predictions and display the respective institutions with which the authors are affiliated.

5.5 Regression Testing

For regression testing we established a DevOps pipeline in GitLab that was able to test the project after any changes and before the modified system was deployed. This gave our project the ability to ensure that any enhancement or modification did not regress the project backwards. We implemented tests to ensure our connection to the database and model remain functional since those are very essential. We also have regression testing to ensure that all our pages remain active to users. We used Pytest for these tests.

5.6 Acceptance Testing

When it comes to functional and nonfunctional requirements being met, we directly involved the client. The acceptance of the product was decided based on criteria such as: intuitiveness of the UI, efficiency of the overall response time, effectiveness in terms of the accuracy of the model. Also, we involved the client in the different stages of integration testing too. Based on feedback from the client, more testing was added, and the final design was accepted.

5.7 Security Testing

While we recognize the importance of security and privacy requirements, these are not the objectives of this project. However, we did do some basic testing on the UI to mitigate SQL injections. These tests made sure that users were not able to directly input SQL into a predictive query to cause an injection.

5.8 Results

Our unit tests and regression tests ran every time we updated the system to ensure continued functionality. Our client has approved of the state of and appearance of our project at this time. All bugs that we ran into while testing have been squashed. For example, we found a problem when a paper had HTML code in the title, but that was later fixed. All tests currently perform to a satisfactory level.

6 Implementation

Our implementation process was to start by adding basic functionality to our project and to gradually implement more complex features that link together to create our cohesive project. This allowed us to separate the problem into smaller chunks including: project infrastructure, a map implementation on the website, user queries, and the ML model. These chunks were expanded upon such as: more features for navigating the map, forward geocoding, and training the ML model. Then these chunks were linked together. This process allowed us to work in an agile methodology with simultaneous development.

There were several areas of development that contributed to our final project. The first area was the acquisition and setup of our project infrastructure. This includes the two servers used in the project, development operations (DevOps), and database management. The first step for implementing our infrastructure was to acquire a development server from the Electronics and Technology Group (ETG) at Iowa State University. Our team obtained a server "sdmay23-35.ece.iastate.edu" that was able to host our Dockered Flask application and our MySQL database. We also acquired a server "sdmay23-35-pc.ece.iastate.edu" from ETG that hosted our ML model. This server was equipped with a Nvidia GTX 1080 which was needed for the HINTS ML model, as Cuda 10.0 was required. Using these servers, we were able to implement DevOps which took the form of a Continuous Integration/Continuous Deployment (CI/CD) pipeline in GitLab. This pipeline significantly aided our implementation as it allowed for changes made to the website to be automatically deployed on the server. This pipeline was also responsible for testing the project before a new version was deployed. Docker was also used to contain our web application with its required dependencies. Flask was also set up with endpoints for the website. Lastly, database management was responsible for getting permissions to use the APS database and adding the data to our MySQL database hosted on "sdmay23-35.ece.iastate.edu". This posed a challenge when the APS data was not uniformly formatted, so a scraper script was created to properly format the data and add it to the MySQL database.

The next area of development was the geo-spacial visualizations on the website that were completed using Mapbox. This implementation started out with basic functionality and progressed with several important features being added. Basic features that were implemented first was the ability to add custom markers to specific latitude and longitude. Later we added the ability to forward geocode, which is the process of getting the latitude and longitude from a location name, using Mapbox. Other significant features that were implemented were aesthetic and usability. This included using CSS and HTML to stylize the map and create auto-zoom features. Lastly, we implemented linking the queried paper to its affiliations and their locations to the map. This functionality was achieved using MySQL queries and linking the paper meta-data to the forward geocoding which gave the latitude and longitude needed for the placement of the markers/pins.

Another area of development was implementation of the creation of user queries. This was implemented in HTML on the Flask application. Implementation of this area was relatively straightforward as it provides input for users to customize their prediction query. Significant features added to this area was an autocomplete feature when looking for papers to generate predictions. The textbox is able to autocomplete papers that are stored in the MySQL database using SQL queries, HTML, python, and CSS. This area was linked to the map visualizations by providing the associated web pages with a list of locations of affiliations from the imputed paper.

The last area of development was implementation of the ML model. This was the most challenging aspect of the project because we were not in charge of the development of the ML model or its training. During implementation, we enlisted the help of a graduate student to aid in our implementation of the ML model. We ran into several issues with our infrastructure requiring another server to host the model. For the current implementation of our design, we have stored the data from running the trained model on the APS database on that server. We are then pulling a time series from the stored data and displaying it to the user. Future work on our project would be to run the trained model every time a user selects a paper and parameters.

6.1 Evolution of Design

In this section we note the changes that our project encountered during implementation and why they were needed. We respectfully note that the images presented in this document do not reflect the team's previous design document. This is because of challenges that arose during implementation causing changes to our project design.

Our original design planned to utilize the React framework for our frontend. After our initial testing, we realized that using React would have added too much overhead to our project, and reduced performance. Vanilla Javascript provided all of the functionality we needed, so it did not make sense to port our design over to React.

We also originally planned to have multiple information cascade models setup for this project, but due to time constraints, and compatibility issues, we ended up sticking to only using the HINTS model provided. We also had to switch to using a server with an older GPU (Nvidia GTX 1080 v.s RTX A6000), as the newer graphics card did not support the model we were using, as the old model relied heavily on Tensorflow version 1.14, which requires CUDA 10.0 to function. Additionally, we planned on having multiple databases for a user to choose from. With this, a user would have been able to train the different models on various databases using the website. This was changed to reflect a proof of concept design requested by the client.

With the above changes, we ended up having to use 2 different servers. We used an Ubuntu VM server in order to host our Flask docker container, as well as our MySQL server. We had a physical server that was used for the training of our model, and for our prediction functionalities.

7 Closing Material

7.1 Discussion

We made a proof of concept product to assist students and professors in developing their research ideas and gathering information and funding for research projects. We have met the design requirements given to us by our client.

7.2 Conclusion

We have successfully completed the proof of concept design and implementation for our project. Testing ensured the functionality of our product and the team communicated with our client to validate the product. Our project was able to utilize the model data for creating predictions and to accurately geo-specialty display paper affiliations alongside these predictions. For implementation we started by implementing basic functionality such as a basic map implementation, we then added more complex functionality such as getting the locations of paper affiliations through forward geocoding, and so on until we had a fully implemented design. This allowed us to break apart the problem that is proposed by the project. Smaller problems allowed us to get the work done more quickly and efficiently and allowed us to test each part of our design.

7.3 References

X. Chen, F. Zhou, K. Zhang, G. Trajcevski, T. Zhong and F. Zhang, "Information Diffusion Prediction via Recurrent Cascades Convolution," 2019 IEEE 35th International Conference on Data Engineering (ICDE), 2019, pp. 770-781, doi: 10.1109/ICDE.2019.00074.

F. Zhou, X. Xu, K. Zhang, G. Trajcevski and T. Zhong, "Variational Information Diffusion for Probabilistic Cascades Prediction," IEEE INFOCOM 2020 - IEEE Conference on Computer Communications, 2020, pp. 1618-1627, doi: 10.1109/INFOCOM41043.2020.9155349.

D. Gotterbarn et al, “Code of ethics: IEEE Computer Society,” Code of Ethics | IEEE Computer Society. [Online]. Available: https://www.computer.org/education/code-of-ethics.

Mapbox. [Online]. Available: https://www.mapbox.com/.

Song Jiang, Bernard Koch, and Yizhou Sun. 2021. HINTS: Citation Time Series Prediction for New Publications via Dynamic Heterogeneous Information Network Embedding. In Proceedings of the Web Conference 2021 (WWW '21). Association for Computing Machinery, New York, NY, USA, 3158–3167. https://doi.org/10.1145/3442381.3450107

Appendices

1. Operation Manual

1. To demo/test our system, please visit: http://sdmay23-35.ece.iastate.edu/
    a. However, our website is only available until the end of Iowa State's Spring 2023 semester, at this point, the servers used to host our website and machine learning model will be wiped.
2. To locally test our web application please clone our git repo: https://github.com/evangossling/sdmay23-35
    a. You must have an adequate version of Python installed. We recommend at least Python 3. Please visit https://www.python.org/downloads/ to install Python.
    b. Once installed, please open the folder in an IDE, we recommend VS Code or PyCharm.
    c. Open the terminal and activate the virtual environment. The command can vary depending on machine and installation.
        i. For Windows:
            1. "`. venv/Scripts/activate`"

```
Evan Gossling@DESKTOP-IT4V0IQ MINGW64 ~/sdmay23-35 (main)
$ . venv/Scripts/activate
(venv)
Evan Gossling@DESKTOP-IT4V0IQ MINGW64 ~/sdmay23-35 (main)
$
```

Figure A.1.1 Activation of the virtual environment on Windows

    d. Once the virtual environment is activated, please install our requirements. This can be done with one of the following commands (make sure to have pip installed):
        i. "`pip install -r requirements.txt`"
        ii. "`py -m pip install -r requirements.txt`"
        iii. "`python -m pip install -r requirements.txt`"

```
Evan Gossling@DESKTOP-IT4V0IQ MINGW64 ~/sdmay23-35 (main)
$ pip install -r requirements.txt
```

Figure A.1.2 Installation of the required dependencies

    e. Finally, you can start the web application with the following command:
        i. "`python app.py`"

```
Evan Gossling@DESKTOP-IT4V0IQ MINGW64 ~/sdmay23-35 (main)
$ python app.py
 * Serving Flask app 'app'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:80
 * Running on http://10.30.199.174:80
Press CTRL+C to quit
```

Figure A.1.3 Launching the web application

f. Now the web application is running on **127.0.0.1/** or **localhost/**

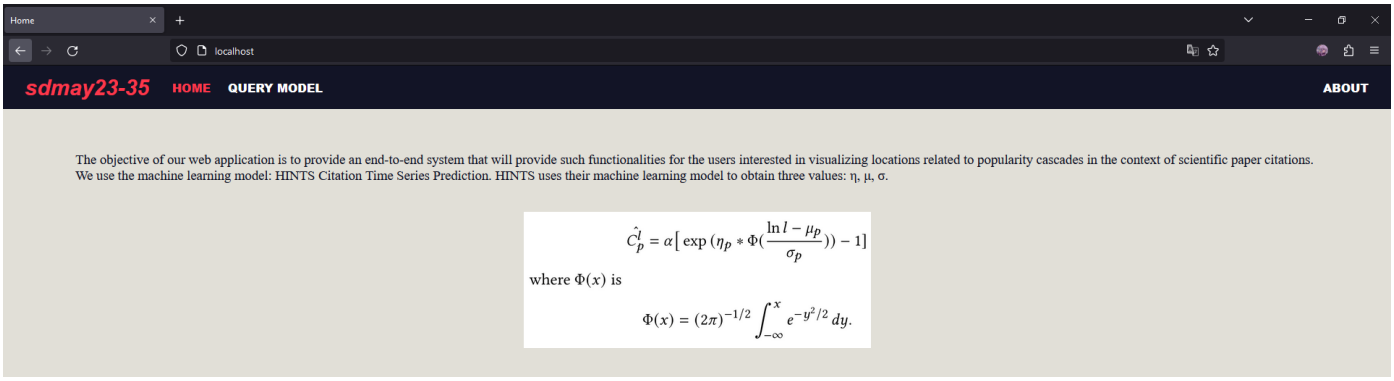

Figure A.1.4 The running web application

g. At this point, you are now able to demo our web application, however our some functionalities will be limited as our database storing our papers, as well as our machine learning model were hosted on servers no longer available.

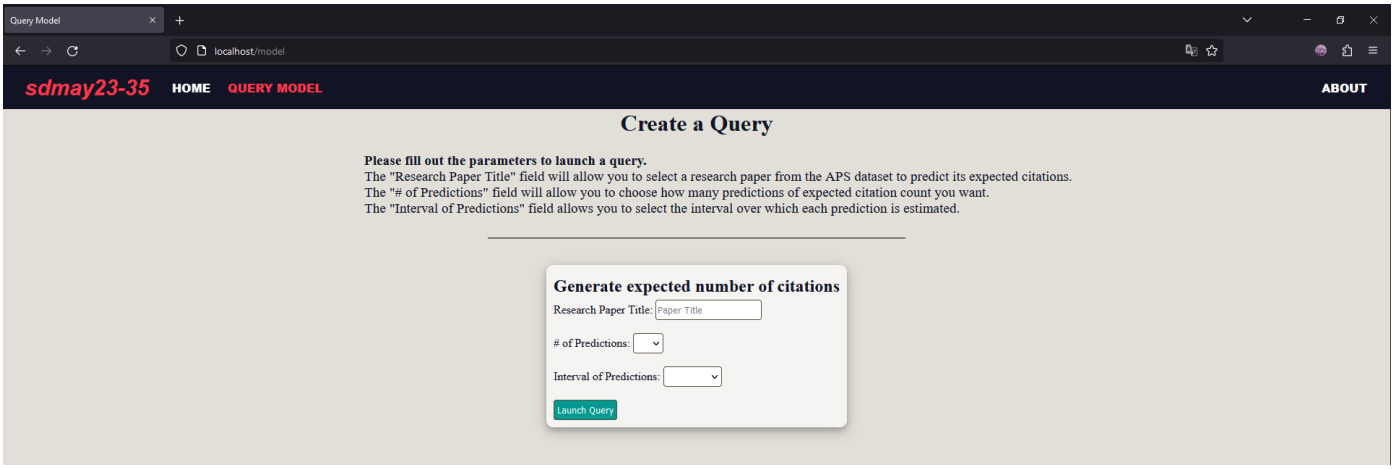

Figure A.1.5 The query creation page of the application

h. To launch a query enter the provided information into the textboxes and click ”Launch Query”. This will transfer to a viewing page after the prediction is executed.

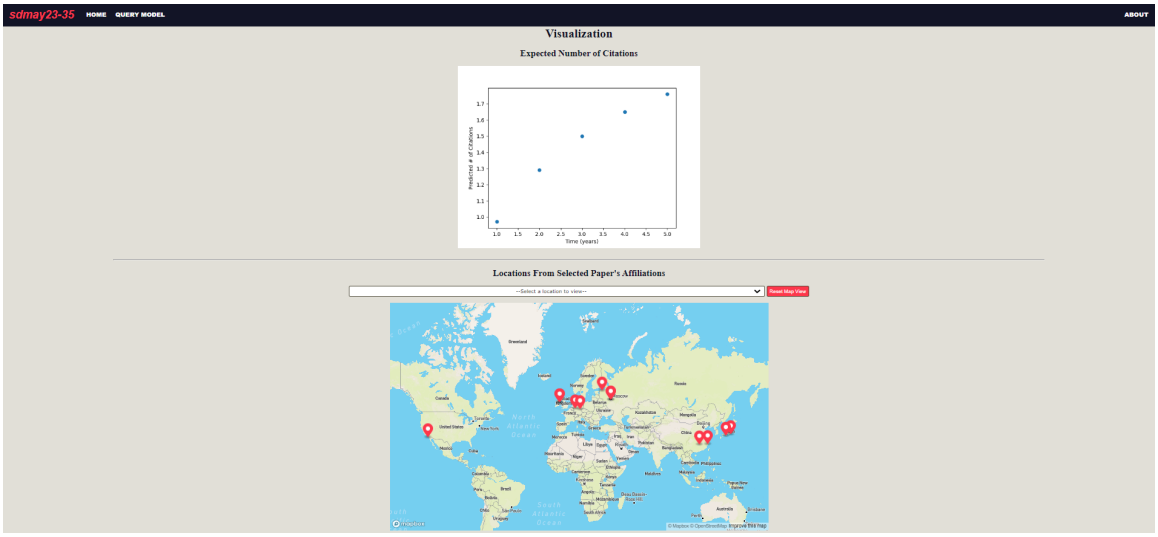

Figure A.1.6 The visualization page of the application

i. Once on the visualization page, one can view the results of the model prediction by analyzing the provided graph. One can also see and interact with the affiliations of the citations using the map. The map is able to be interacted with by zooming in and out and by clicking on the markers/pins to quickly focus on a location. One can also focus on a location by selecting it from the list in the dropdown.
j. To stop the application go back to the terminal and **Press CTRL+C to quit**

3. In order to setup the database side of the application, a MySQL database must be setup (or other similar architecture, i.e. MariaDB).
    a. MySQL server setup can be acquired at https://dev.mysql.com/downloads/mysql/.
    b. Please follow these MySQL setup instructions: https://dev.mysql.com/doc/mysql-getting-started/en/
    c. The APS dataset needed for setup can be found at https://journals.aps.org/datasets
    d. After the dataset has been acquired, on the server where your MySQL server is located, run the following script in order to import the data and create the needed tables.
        i. dataset_parser.py (this will run create.sql & drop.sql)
    e. In __init__.py the server configuration settings must be changed to the values of your own server.
        i. app.config['MYSQL_HOST'] = $HOSTNAME
        ii. app.config['MYSQL_USER'] = $USERNAME
        iii. app.config['MYSQL_PASSWORD'] = $PASSWORD
        iv. app.config['MYSQL_DB'] = $DBNAME
4. The final part that needs to be set up is the prediction model.
    a. The cascade prediction model can be acquired from: https://github.com/songjiang0909/HINTS_code
    b. Follow the steps listed there to setup the prediction model

2. Alternate initial versions of the design

The below images show the initial design of the website's structure and prototyped implementation. This design was scraped to reflect a simpler version of this functionality intended to be a proof of concept for our client. For a list of adaptations and design changes see Section 6.1.

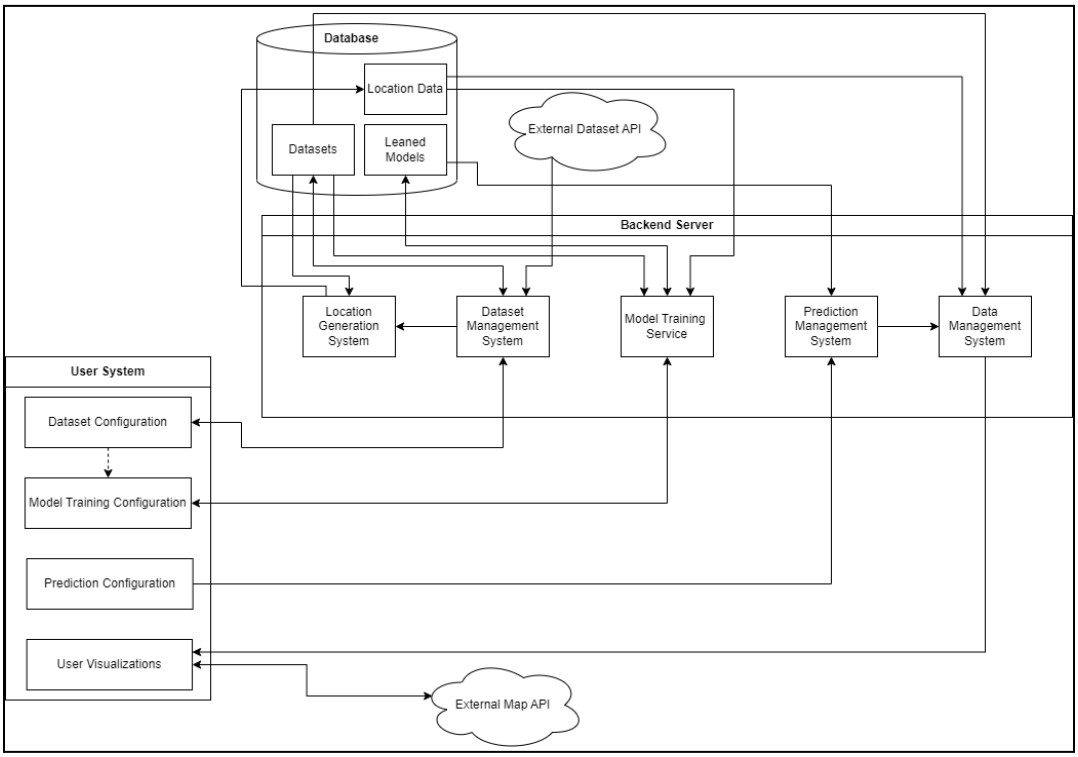

Figure A.2.1 Original Architecture

Below in Figure A.2.2 and Figure A.2.3 you can see the initial design of our Model Creation and Prediction pages, we have since then switched to the implementation as seen in Figure 4.4 and Figure 4.5. We made these changes after we gained a better understanding of what the Client wanted and what was required to fit the project specifications.

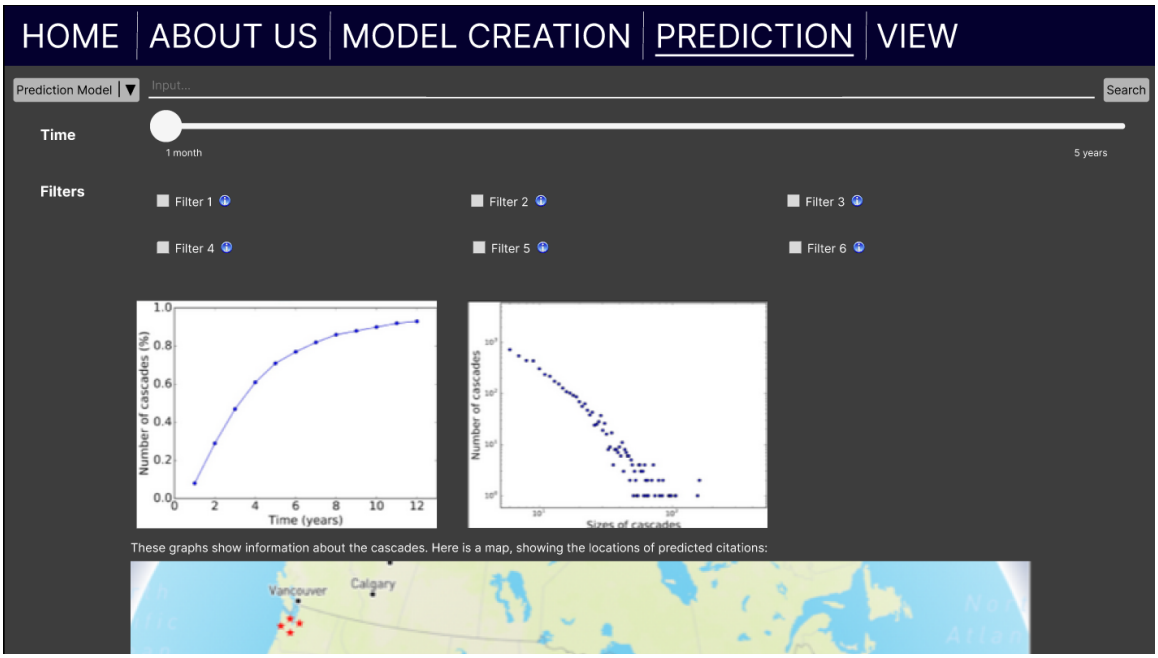

Figure A.2.2 Original Web Page Design

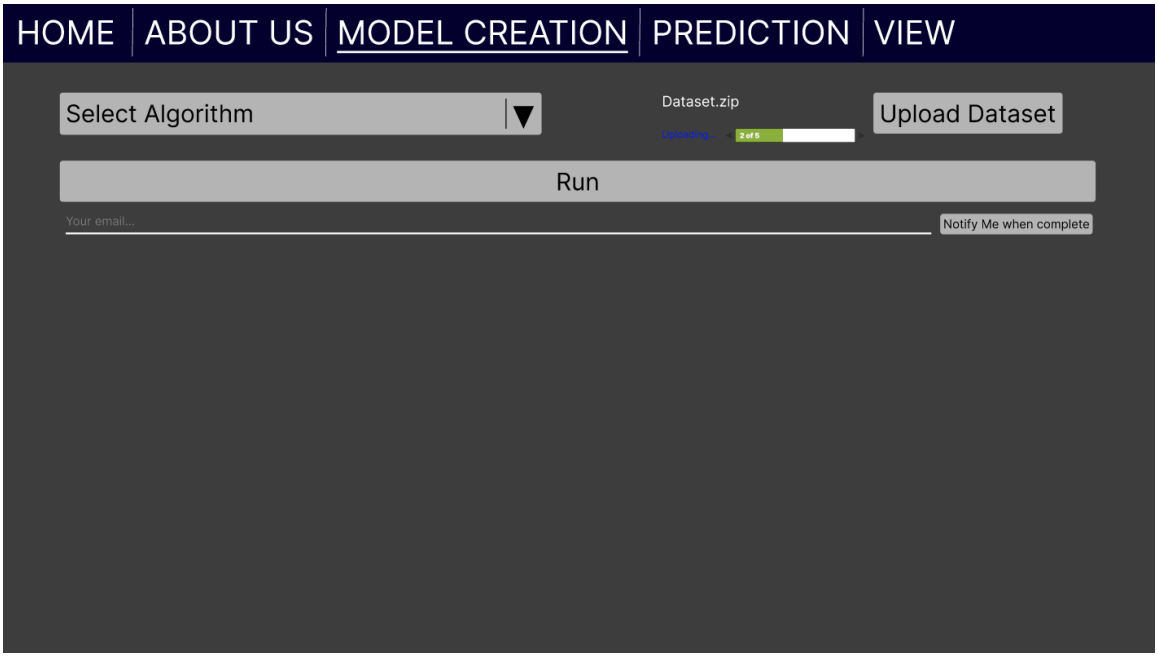

Figure A.2.3 Original Web Page Design

3. Other Considerations

Something that our team wanted to mention was the help and support that we obtained from Ce Le in implementing the ML model. Our team reached out to Ce to aid in our understanding about ML models that was lacking during our initial design. Ce was able to guide us through training and querying the HINTS predictive algorithm and was a valuable resource to the team.

4. Code

To view our code, please visit: https://github.com/evangossling/sdmay23-35