Censorship Detection

John Doros
john.doros10@bcmail.cuny.edu

Supervisor <Professor Chen, chen@sci.brooklyn.cuny.edu>

Abstract

Censorship detection is a ambitious challenge in understanding internet freedom as organizations use various forms of content restriction and blocking policies. This project aims to focus on preprocessing and data aggregation on censorship measurement data from Censored Planet to achieve a more accurate form of detecting censorship events.I will focus on processing raw censorship measurement data integrating metadata to it and applying aggregation methods to identify patterns in censorship behavior. My goal for this project is to develop an efficient data processing method and or approach that enables to identify potential censorship events while differentiating from false positives for example such as network outages. The understanding and methods developed from this project will contribute to a broader approach in efforts for censorship circumvention and or reliability in the future internet. This project will focus on supporting the CenDTect framework focusing on step 1 of CenDTect Architecture which is Raw data preprocessing, which transforms censorship measurement data into a more suitable structured format for a better approach to censorship analysis.Throughout this project the main goal is to advance the effectiveness of censorship detection methods and to contribute to a more reliable internet.

Research Focus & Hypothesis

This project focuses on how censorship measurement data can be processed and aggregated to improve the accuracy of detecting censorship while reducing the amount of false positives. By analyzing a large Censored Planet dataset I aim to develop a structured approach for classifying censorship events and be able to differentiate between actual censorship events and non censorship events such as network failures.

Research Question

How can censorship data be effectively aggregated and preprocessed to improve censorship detection accuracy while reducing false positives?

Hypothesis

If censorship data is preprocessed with aggregation techniques and with metadata integration then censorship events can become more accurately identified while reducing false positives.

Background on Censorship Mechanisms

Background on censorship mechanisms.

DNS Blocking: Preventing domain name lookups such as failing or redirecting

IP Blocking : Preventing connections to a IP address

HTTP/HTTPS Filtering: Blocking specific content on webpage

Network Disruptions: Packet injections or network outages

Previous Research in Censorship Detection

OONI : Volunteer based censorship measurement gathering organization

Censored Planet : Censorship measurement database using global vantage points

CenDTect : Decision tree based censorship event detection

Research Methodology

Data Collection

Extract censorship data from Censored Planted

Preprocessing and feature extraction

Cleaning the extracted data and extract metadata (ASN, HTTP response codes, ISP, TLS details)

Classification and testing

Categorize response as normal censorship, likely censorship or unlikely censorship, compare these detected events with OONI reports for validation

Aggregation and trend analysis

Grouping data by ISP, country or censorship type, and assess false positives (such as network outage)

Finally measure accuracy of censorship detection (by comparing with ground truth)

Tools

GitHub - Used for uploading all project related files, used for backup as well.

SQL - Used for storing and querying censorship data efficiently supporting data aggregation analysis.

Dropbox - Used for retrieving censorship data provided by my supervisor.

PySide2 (Python Library) - Used for creating simple interactive interfaces for visualizing censorship analysis if needed.

Matplotlib - Used for visualizing censorship analysis if needed.

OONI/Censys - Has real world censorship data trends used for cross checking my own data set to determine its accurateness.

Python - Programming language used for data processing/validation/aggregation makes it easier for data filtering

Data Sources

The data sources that will be used are as followed

Censored Planted “https://censoredplanet.org/” - Used for extracting raw censorship data measurement (Professor has provided a dropbox link where he had extracted this data so I can use it)

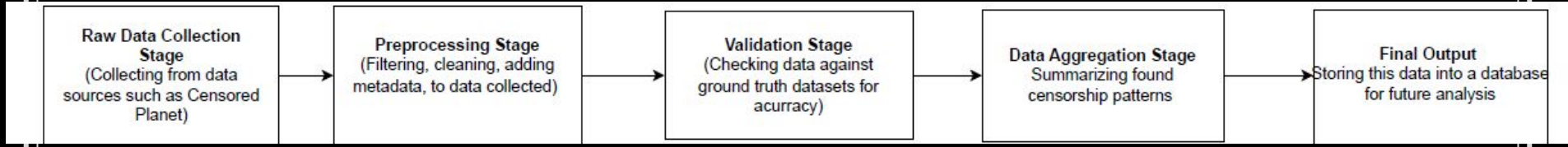
OONI “https://ooni.org/” - Possibly used for cross checking/validating censorship events for further accurateness

Citizen Lab “https://citizenlab.ca/” - Can be used for further checking/validating censorship events

ICANN “https://www.icann.org/” - Can be used for metadata extraction on IP address

WHOIS “https://who.is/” - Can be used for metadata extraction on IP address

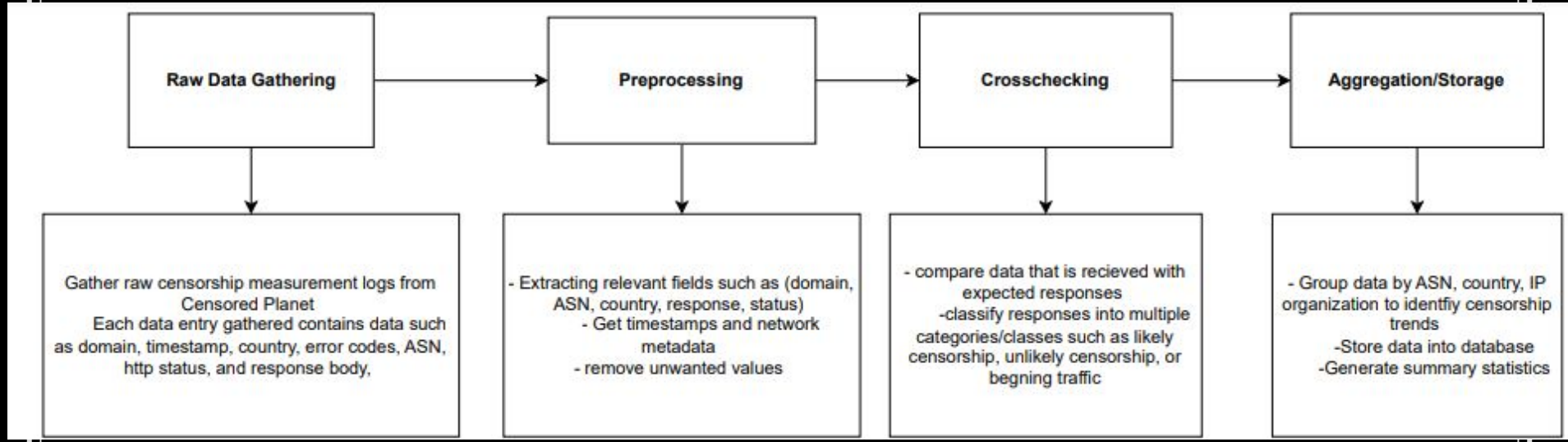
Stage Flow Diagram For Data Processing

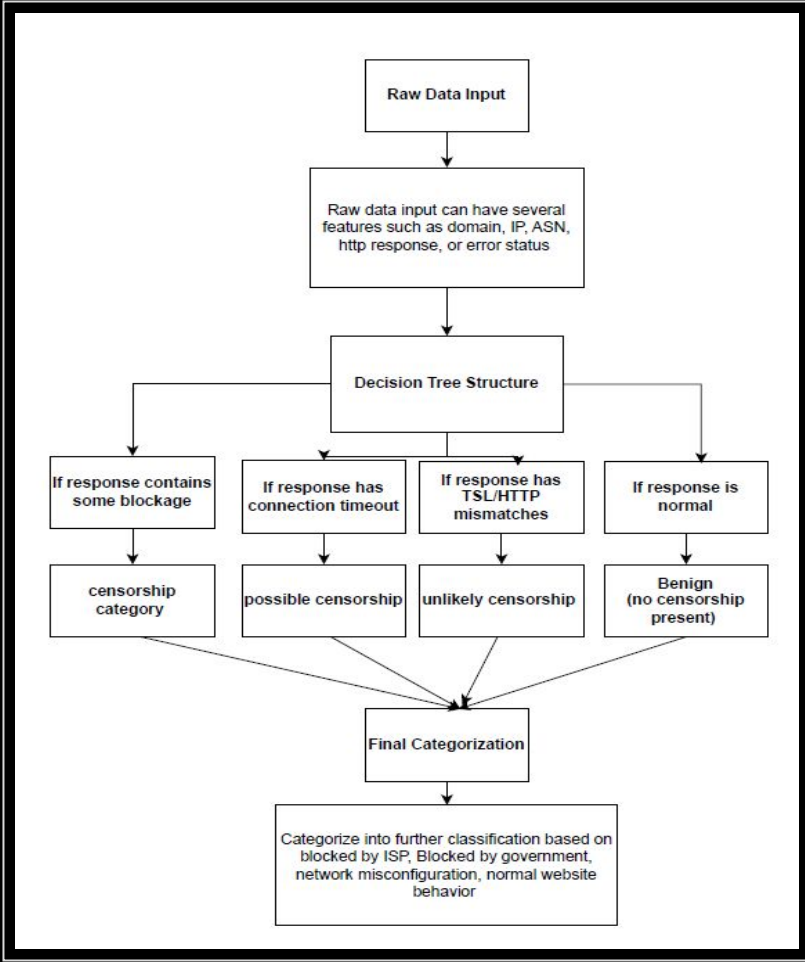


Showing the data processing states at each stage and what will happen to the raw data at each stage.

Logical flow of data processing (Sequence of operations on data)

Showing the what happens on processing pipeline in detail





Censorship event classification

How is a censorship event classified based on a decision tree flow , this would classify events based on different conditions, useful for demonstrating how events are analyzed to determine if censorship is likely or not into different classes

Use Cases

Some use cases for this technical implementation can be as follows

Identifying government website blocking : Governments in some areas block access to website contents to which can include news, social media and other websites to which governments do not like people to access at a particular time. This is important as people in society would form a bias towards the government and whether the government should be trusted or not.

Distinguishing between network outage and real censorship events : When it comes to detecting censorship events some events can create false positives to which the access to certain content are not caused by a censorship event but rather a network failure and or outage. This use case can minimize those false positives by distinguishing from a network error and a censorship event, this can have an impact on our data that we are processing to which can make our data more reliable and more accurate when it comes to detecting a censorship event.

Tracking censorship events over time : When it comes to a censorship event taking place such as during a political event, protests and or elections tracking censorship during these periods can be helpful for monitoring censorship trends over time to detect patterns and obtain further analysis. This can be useful to push for transparency during these periods and further can help detecting new censorship events and patterns.

February
Generally: Work on required documents presented on Blackboard, setup meetings with supervisor for project discussion

Feb 19. Reformat slides based on Supervisors meeting including a log file
Estimate time : 1 week

Feb 20.
Start updating on project management dashboard
Estimate time : 1 day

Feb 20-28. Start working on data filtering software
Estimate time until completion : 1 month

Organize version control repository and work on required Blackboard documents
Estimate time : 1 Week

First week of March:
Prepare and work on demos
Estimate time : 1 week

Continue to stay in touch with supervisor
Weekly basis

March:

Week 1 : Finish working on software
Estimate time : 1 week
Week 2 : Work on bug fixes in the software
Estimate time : 1 week
Work on deliverables required on Blackboard
Estimate time : 1 week

Rest of semester :
Continue to work on software, continue updating supervisor, continue to update project management tools
Estimate time : couple weeks

Tentative Schedule

Will continue to update this schedule but this is a rough idea for now

Use Cases

Some use cases for this technical implementation can be as follows

Identifying government website blocking : Governments in some areas block access to website contents to which can include news, social media and other websites to which governments do not like people to access at a particular time. This is important as people in society would form a bias towards the government and whether the government should be trusted or not.

Distinguishing between network outage and real censorship events : When it comes to detecting censorship events some events can create false positives to which the access to certain content are not caused by a censorship event but rather a network failure and or outage. This use case can minimize those false positives by distinguishing from a network error and a censorship event, this can have an impact on our data that we are processing to which can make our data more reliable and more accurate when it comes to detecting a censorship event.

Tracking censorship events over time : When it comes to a censorship event taking place such as during a political event, protests and or elections tracking censorship during these periods can be helpful for monitoring censorship trends over time to detect patterns and obtain further analysis. This can be useful to push for transparency during these periods and further can help detecting new censorship events and patterns.

Expectations and Future Work

-Research expected contributions

Develop a structured data processing approach for detecting censorship

Created an improvement to reduce false positive

Provide trend analysis on censorship patterns over time

-Future considerations

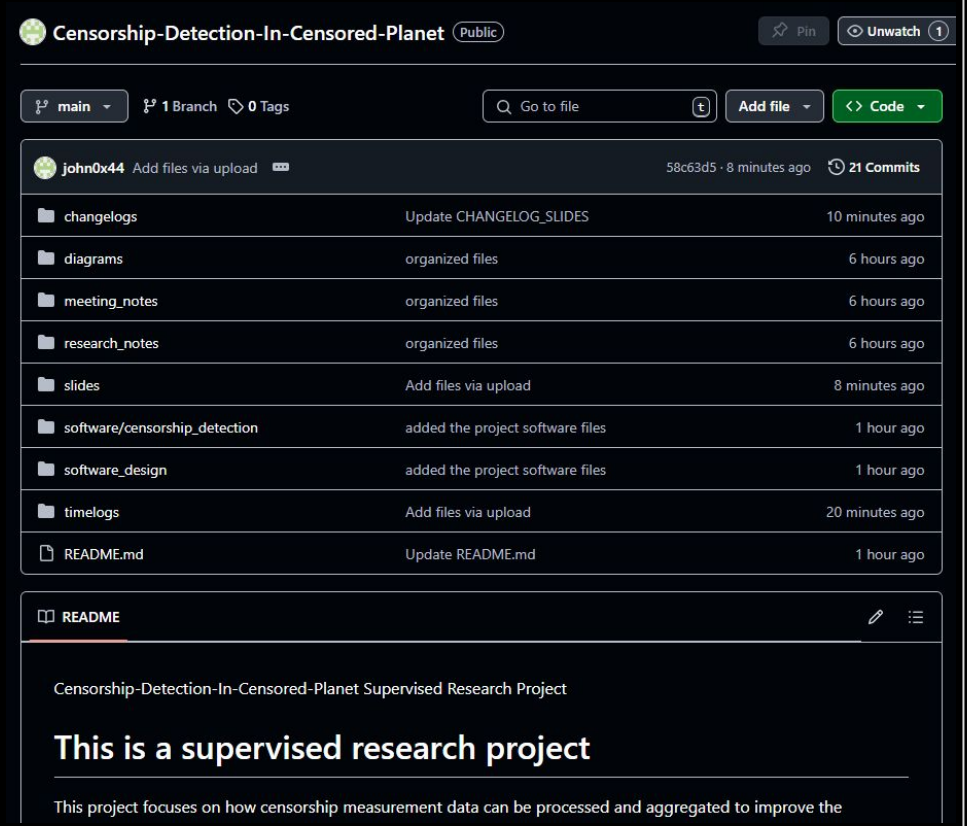
Using machine learning models for censorship classification

Improve accuracy through combination of multiple datasets

Expand to focus on detecting new types of censorship

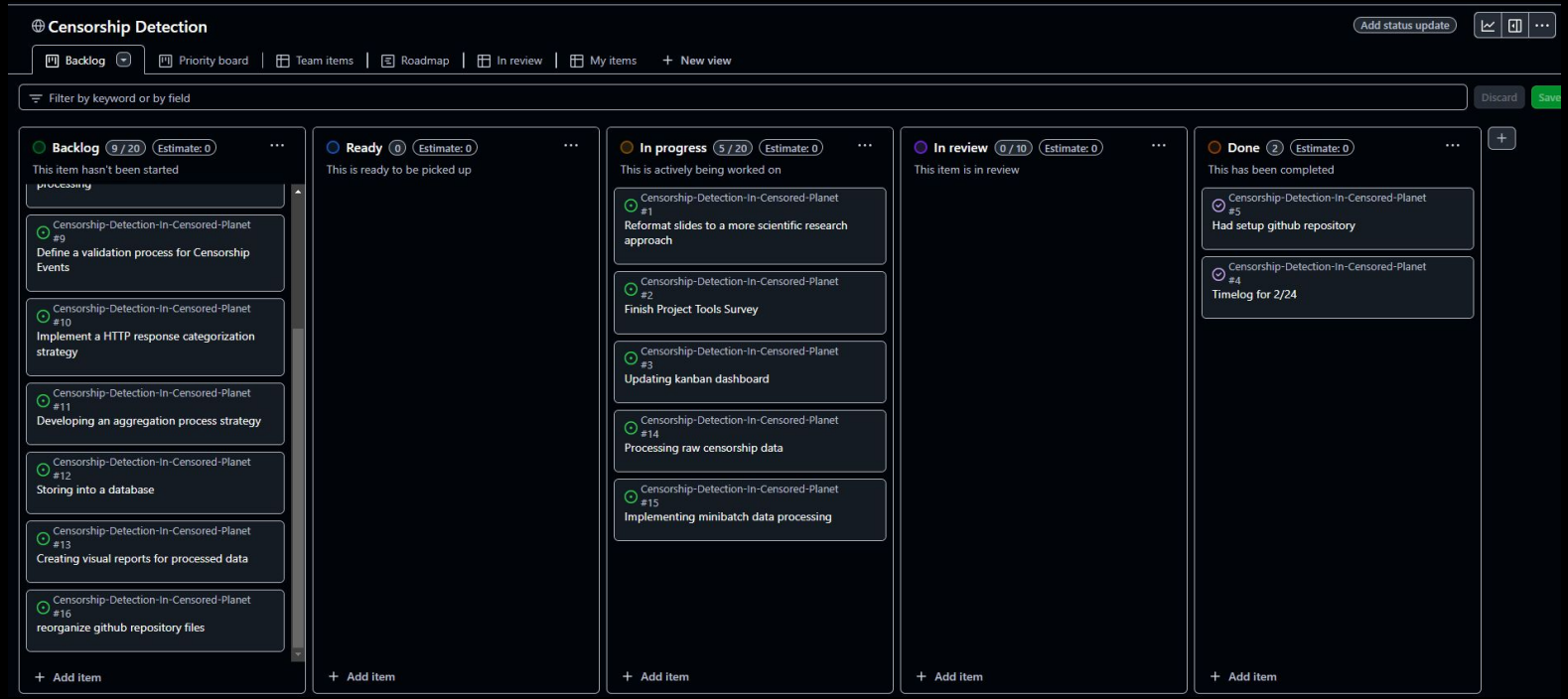
Github Repository

https://github.com/john0x44/Censorship-Detection-In-Censored-Planet/tree/main

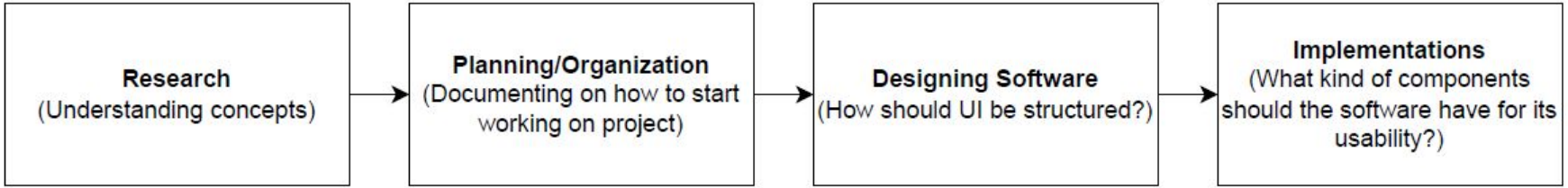


Project Management

https://github.com/users/john0x44/projects/1/views/1



Evolution



Started the project by researching and understanding concepts that were related to the censorship detection research report. Next was to plan and organize how project materials were to be organized and how I should start working on the project itself. The planning portion was about first about designing an intuitive interface for the software and how components should be laid out onto the UI. In the implementation phase I had to connect and work on components that were necessary for software data processing/usability which is the current phase.

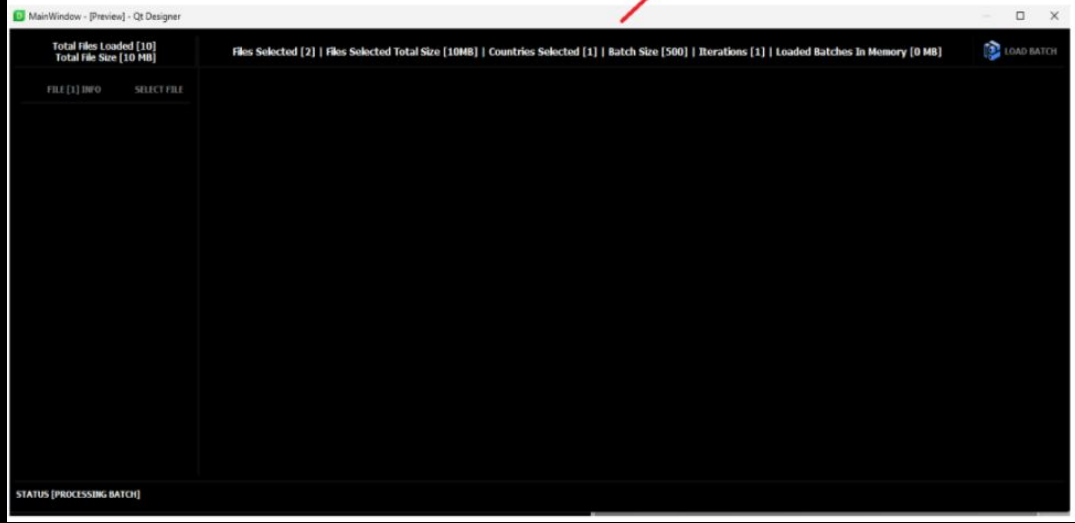
Research Phase

In the research phase I had to understand concepts that were presented in the “Cndetect” research paper, some concepts that I researched were what types of censorship events occur such as for example on the DNS level, TCP level/Http level. Why are there challenges in validating censorship events as organizations do not report on censorship policies so validating censorship events is a difficult challenge. How different censorship detection methods can lead to false positives as there can be a detection on censorship while there is no actual event happening leading to a false positive such as utilizing time series based anomaly detection, or stationary data analysis. How “Cndetect” utilizes their own methods for censorship discovery by utilizing decision trees to find blocking policies in events and then are organized into different censorship categories. And most importantly how “Cndetect” utilizes their approach in the aggregate measurement process in order to process raw data more effectively, enriching each raw data event with additional metadata to improve detection accuracy which is the focus of this project by implementing this approach in order to classify raw data events as a censored vs uncensored event. The challenge is to come up with the aggregate measurement approach as “Cndetect” does not describe their implementation.

Designing Software

User Interface (Beginning)

The current software UI design making the data processing easily shown to the user which can give us a better understanding of the data we are processing



Designing Software

User Interface (Beginning)

will show what file names did the user add to the software
this will have an about file (showing details of the file)
and select file used for selecting this specific file for the "mini batch" data loading strategy

Some info is shown that will be useful to the user

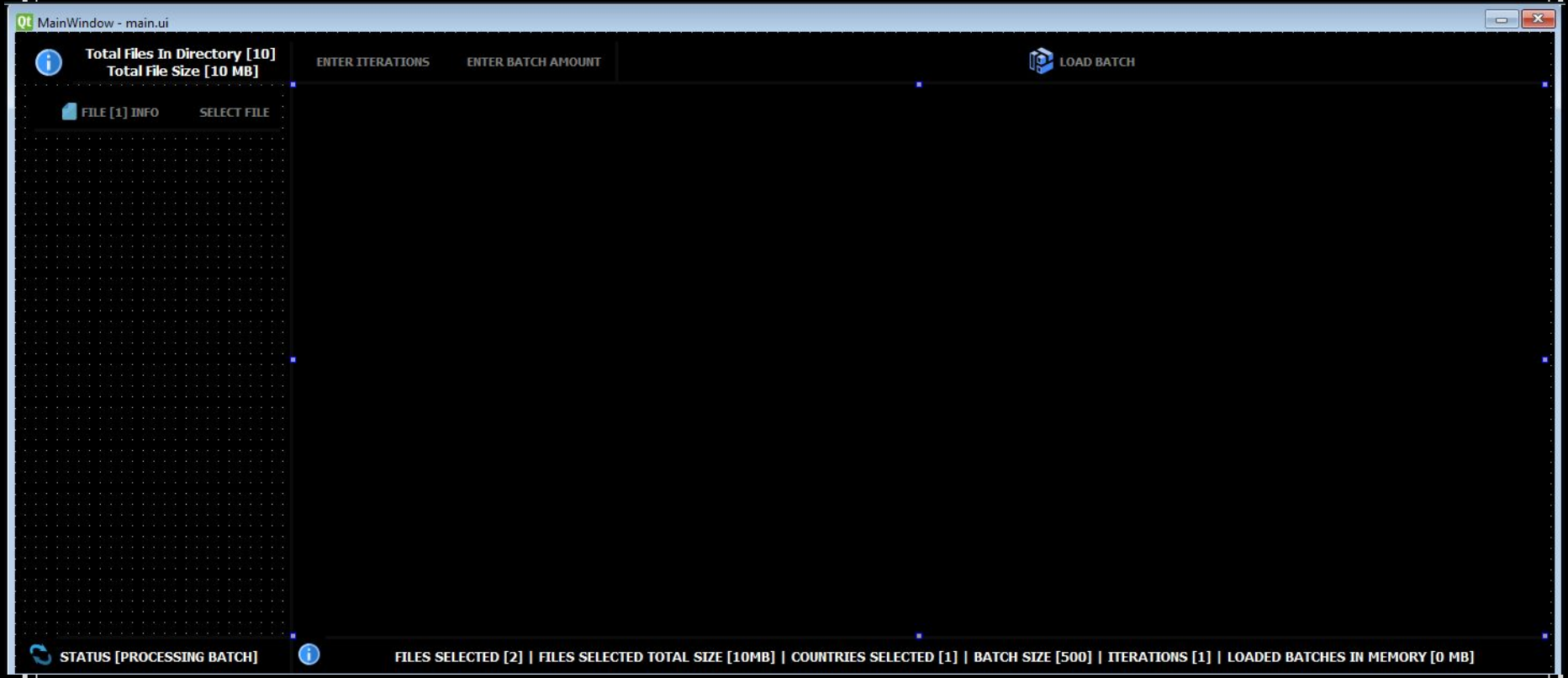
user is able to process the data into memory which will require threading

blank space here will be modified in the future and will include the analysis and other project related data processing pipelines

show the current status of the software

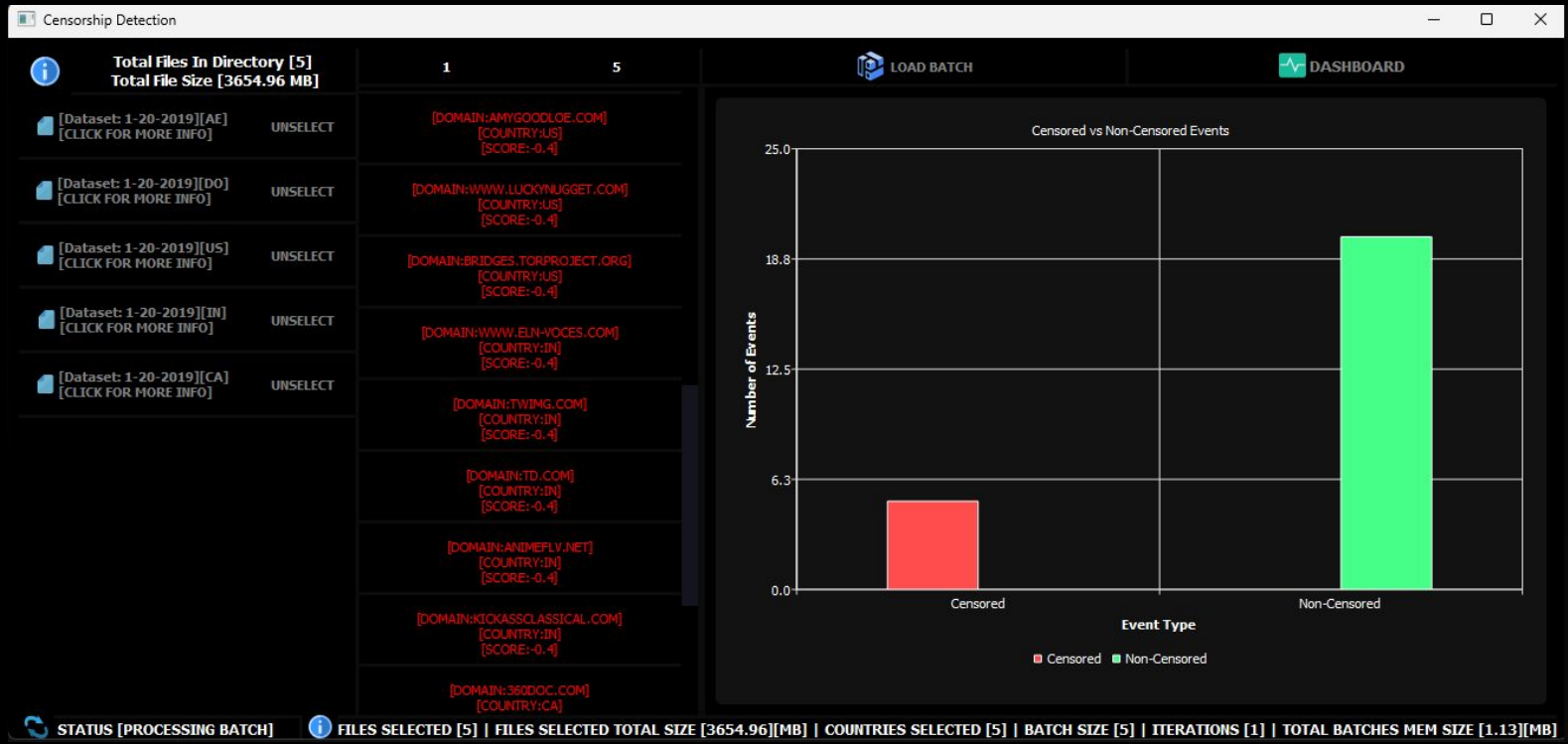
Designing Software

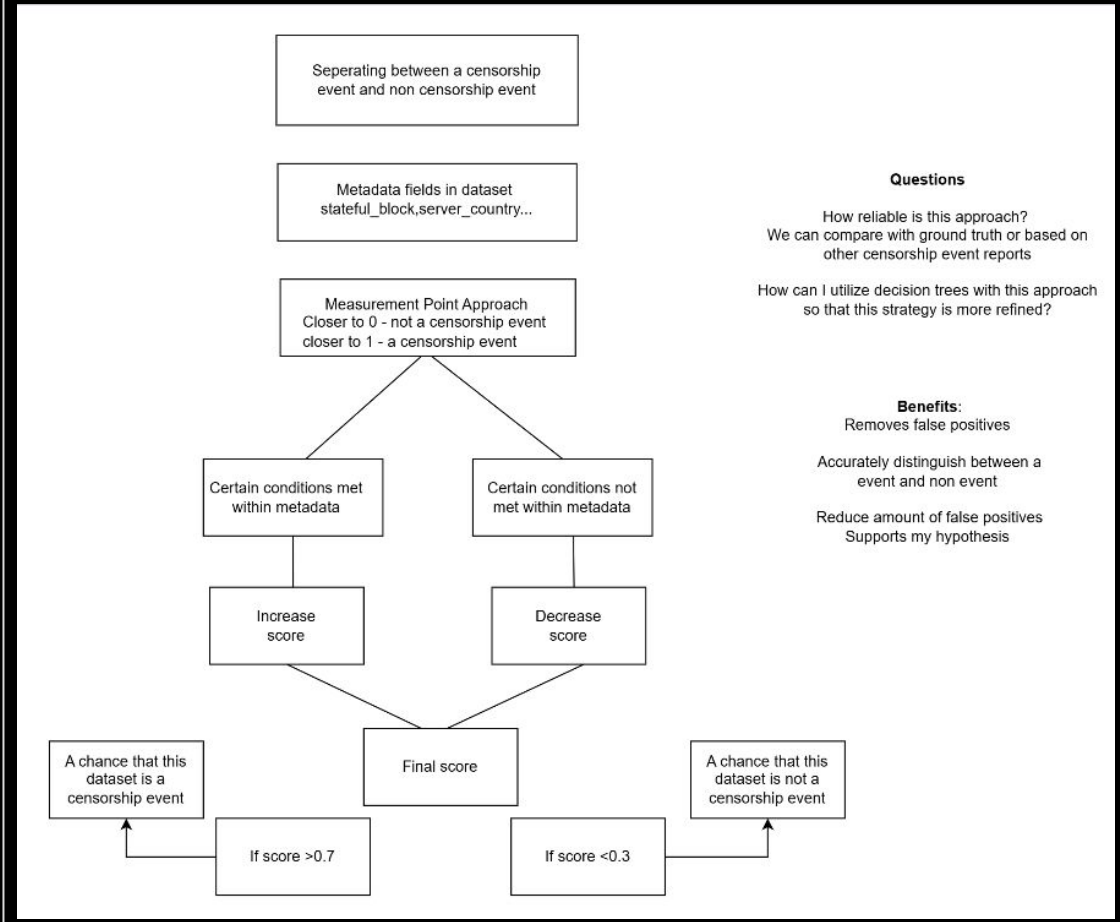
User Interface (Halfway)



Designing Software

User Interface (Current)

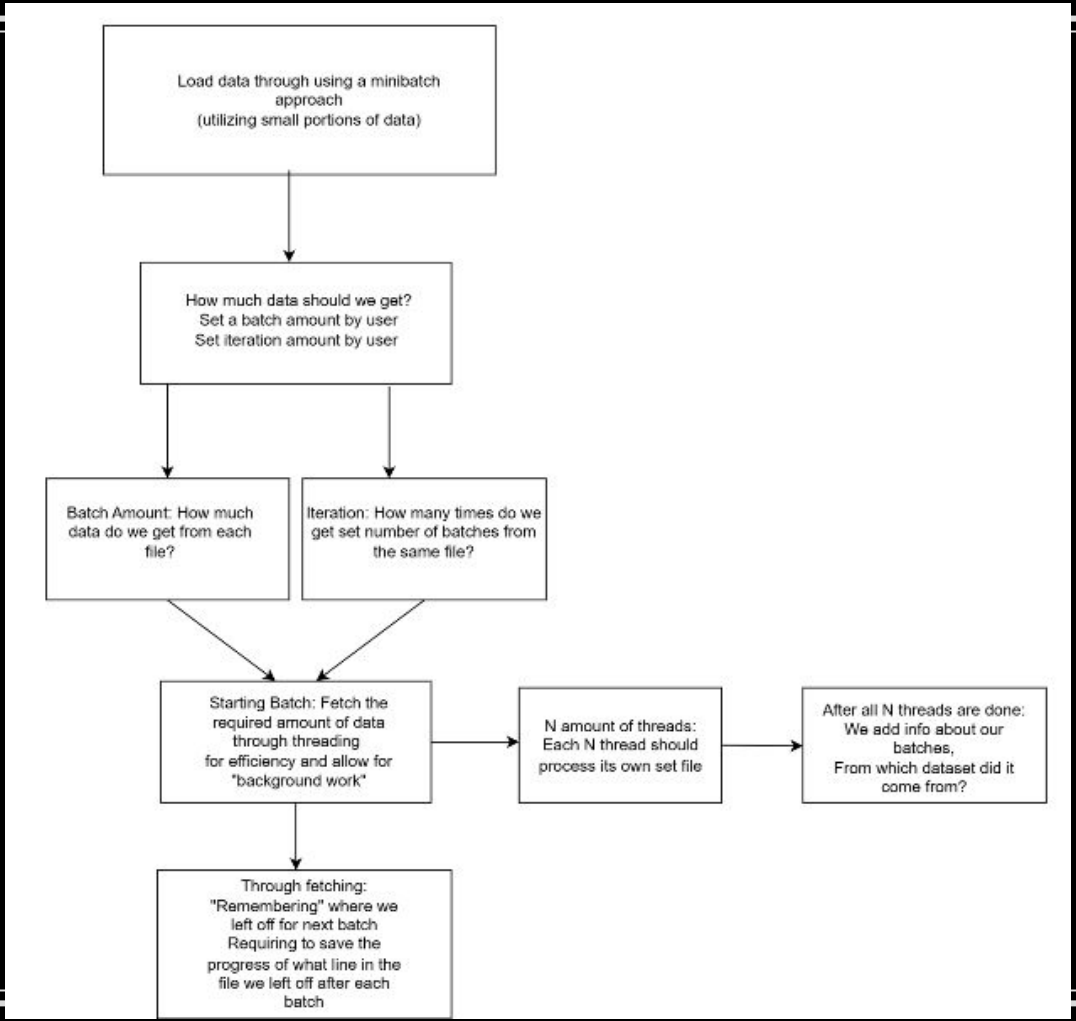




Backend Components

(Example: PointMeasurement Class)

Overview of one of the classes used in the backend of software for example **PointMeasurement** class in which is an implementation of an approach used to determine whether an event is either a censored or non-censored event by utilizing a score for each dataset after examining the metadata fields presented in the dataset each metadata field with a certain condition met can really influence whether this particular dataset is a censorship event or not



Backend Components

(Example: miniBatch Class)

Overview of one of the classes used in the backend of software for example **miniBatch** class in which is an implementation of an approach used to load in dataset data, as the software cannot just load in the entire dataset into memory as the software will slowdown and will not work, so a minibatch approach is implemented and used to let the user select how much data they are willing to process in their machine. The dataset is split into small chunks so only a small amount of data is processed with the help of threading.

MiniBatch Class

(Challenges)

```python
# ProcessBatchThread : Process the batch from each file
# Return the line amount, fileID, batchData
class ProcessBatchThread(QThread):
    Result = Signal(dict)

    def __init__(self, mainDirectory, datasetDirectory, fileName, fileID, batchSize, iterationSize, lastLineReadPosition):
        QThread.__init__(self)
        self.batch = {} #store the batch
        self.mainDirectory = mainDirectory
        self.datasetDirectory = datasetDirectory
        self.fileName = fileName
        self.fileID = fileID
        self.batchSize = batchSize
        self.iterationSize = iterationSize
        self.lastLineReadPosition = lastLineReadPosition # store the last line read position
        self.iterationIndex = 1 #store the batch indexing correctly

    def run(self):
        filePath = f"{self.mainDirectory}/{self.datasetDirectory}/{self.fileName}"
        try:
            with open(filePath, 'r', encoding='utf-8') as file:
                #Move to last read position
                for _ in range(self.lastLineReadPosition):
                    next(file, None)

                for _ in range(self.iterationSize):
                    thisBatch = []

                    for _ in range(self.batchSize):
                        line = file.readline()
                        if not line: #if this is the end of file stop
                            break
                        thisBatch.append(line.strip())
```

One of the challenges during the implementation phase of the software was processing large amounts of data efficiently so we utilize the minibatch approach but at the same time the problem was that the software cannot do other work while it is processing this data, so one solution I had to come up with and through reading documentation was to utilize qthreads in which will help run data processing in the “background” while the software/UI can do other work. So implementing qthreading helped in a sense that the user is now able to do other operations on the software without having to worry about waiting until data processing is complete. We can see this threading implementation in our thread manager class.

Documentation

The most important thing was learning and reading through the QT documentation about designing and working with different interface components that were added to the software interface, such as list views, charts, and how to style the interface itself. Learning and reading documentation as well helped me implement useful services in the backend of the software such as implementing threading and connecting the UI interface with the backend. So the outcome of learning the documentation really helped out in designing and building the software from the ground up. The learning outcome was that I can further improve my ability to program since with programming I can connect components into the backend making a full stack application such as this project.

Documentation used:

https://doc.qt.io/qtforpython-5/PySide2/QtWidgets/index.html
https://doc.qt.io/
stackoverflow.com

Algorithms

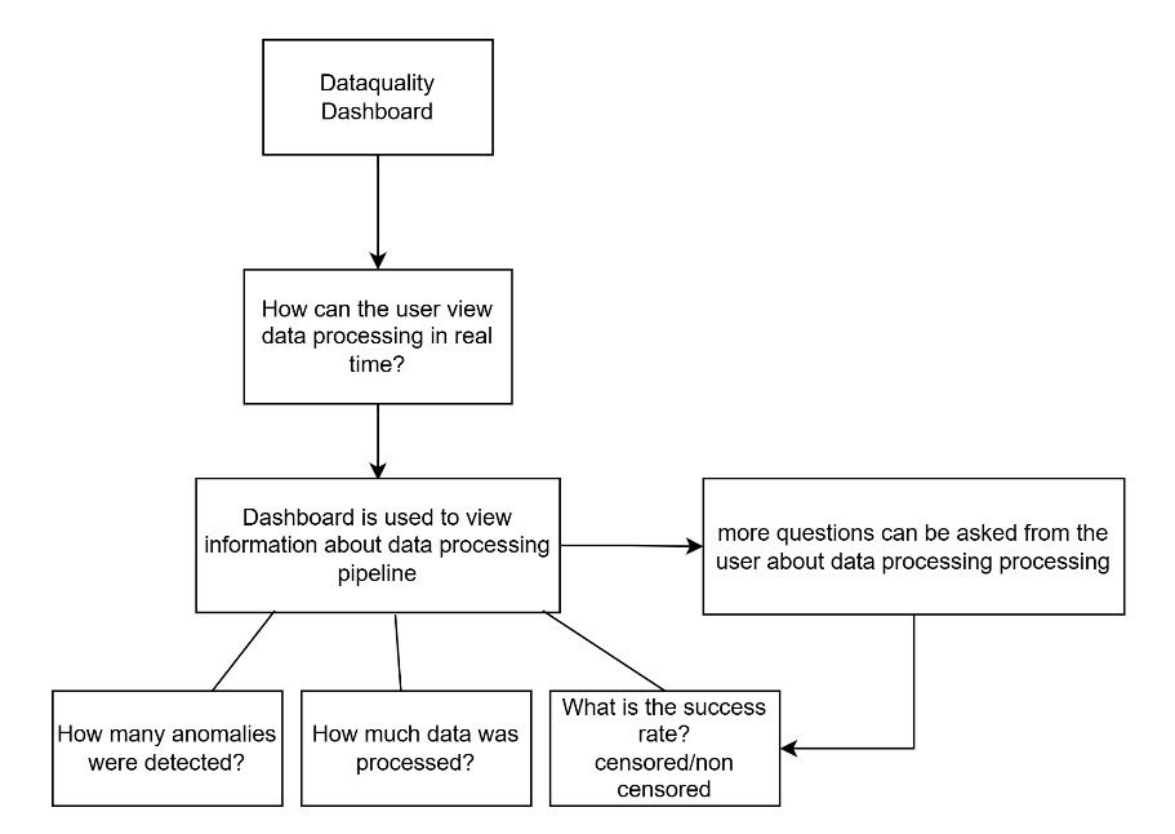
```
def processThreadResult(self,batchResult):
    self.threadsRunning -= 1
    # Before appending the batches we can filter it here
    filteredBatch = self.filtering.filterBatch(batchResult)
    self.currentBatches.append(filteredBatch)

    # we are done with all the threads then we can save the progress of all batches
    if (self.threadsRunning == 0):
        for i in range(len(self.currentBatches)):
            thisBatch = self.currentBatches[i]
            self.dataProgress[thisBatch['fileID']] = thisBatch['lastLine'] #set the last line read from the file in case of future batches processed
            #calculate the batches memory size
            self.batchesMemSize = self.batchesMemSize + (sys.getsizeof(thisBatch['batch'])/1024)
            # with open(self.trackProgressBatch, 'w') as file:
            #         json.dump(self.dataProgress, file, indent=4)
        self.batchesMemSize = round(self.batchesMemSize,2)
        self.updateBatchesMemSize(self.batchesMemSize)

        print(f"Loaded batches in memory {self.batchesMemSize}[MB]")
        self.processingBatch = False
        self.batchesProcessed = self.batchesProcessed + self.currentBatches

        all_batches = []
```

Some components required to develop unique algorithms in order to process the data efficiently, one example is the batch processing algorithm which not only filters out unusable dataset events but also performs final processing logic in a more modular way. This is the core of the software allowing us to easily adjust how the software interprets censorship events. The learning outcome out of this was that I can improve the way I think about implementing algorithms in a way that will be more modular based.



Updated Designs

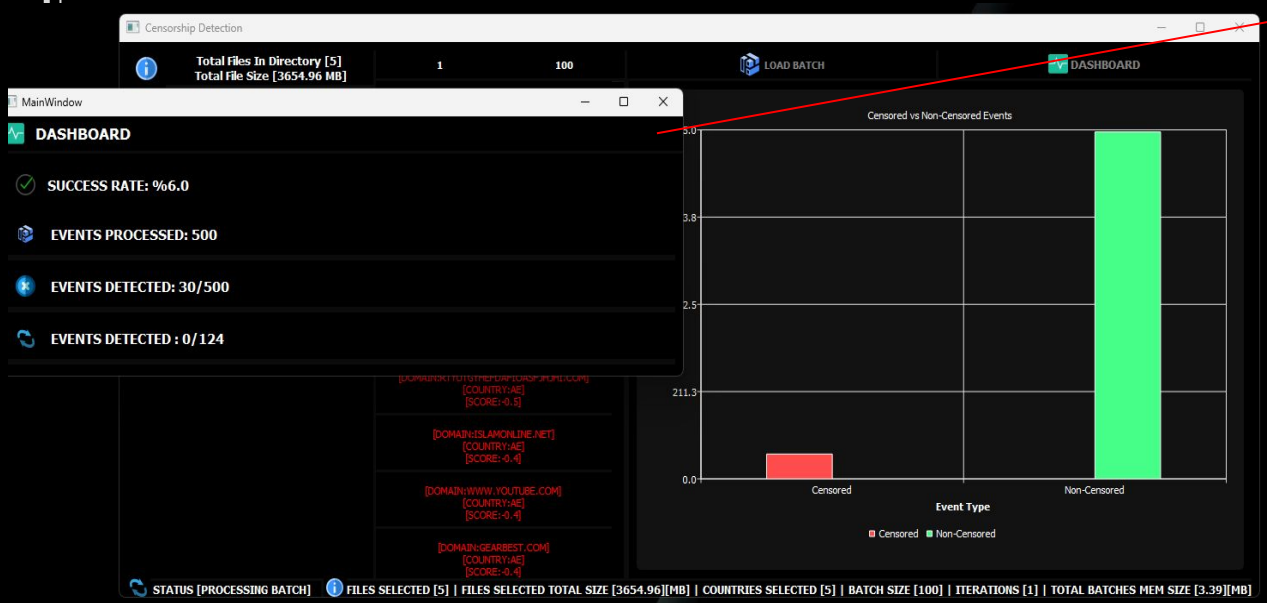
(Dataquality Dashboard)

A new implementation that is used to show the user more insights into the data processing pipeline, some questions that this dashboard would answer is
Were there any anomalies or skipped events?
How much data was processed?
What is the rate of censored vs uncensored events?

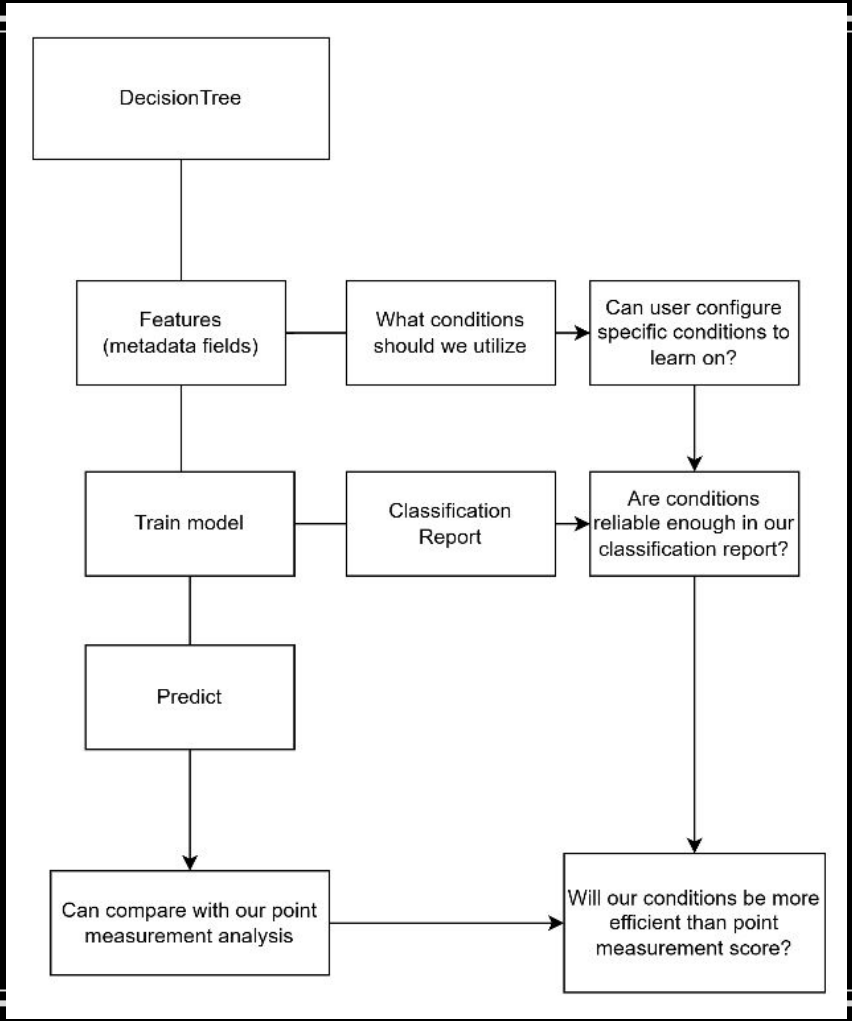
There might be more questions to be answered but we can implement these later for now we should implement an intuitive dashboard that gives the most important insights.

Updated Designs

(Dataquality Dashboard)



We can see this current implemented feature here which I am excited about since we can get more insights into our data that is processed, we can continue working on this feature to incorporate dataset extraction after the data processing is done, but for now it only answers the most important questions about the datasets processed.



Updated Designs

(DecisionTree Implementation)

A new implementation that will use decision trees in order to classify datasets as censored vs non-censored this can help us in a sense that we can compare these results with our point measurement approach and see if one is more reliable than the other. But at the same time we have to figure out a way to prove if our pointMeasurement score is reliable by looking at other censorship event reports, which will be done soon.

Project Details

Github : https://github.com/john0x44/Censorship-Detection-In-Censored-Planet

Project Management Board: https://github.com/users/john0x44/projects/1

Timelog(s) :
https://github.com/john0x44/Censorship-Detection-In-Censored-Planet/tree/main/timelogs

Latest Timelog :
https://github.com/john0x44/Censorship-Detection-In-Censored-Planet/blob/main/timelogs/4900_time_log_week_9.pdf