Project Name: Real-Time Analysis of Microscopic Video Data
Report 3

Project Objectives

Our goal is to stream in yeast, E.Coli,and other bacterial microscopy and tag those cells as well as determine important biological features in real time to help biologists more effectively collect data and experiment. The main object would be to develop a lineage model of bacteria between frames as well as determine and tag significant events, such as cells splitting or interacting or dying. This could also provide real time feedback about the window quality of the microscope on the sample, allowing the biologist to move the window to get a better view(such as making sure a minimum of multiple phenotypes are within the region depending on the experiment.)

Significance

Cell microscopy and observation is largely done by specialized human workers as humans are the gold standard of image detection and recognition as well as knowing what important biological features to assess. This is costly as the person's time could be used on other facets of research and human labor itself is costly.

Cell experiments are also performed in stages. That is, one single gene or biological factor is looked for and analyzed through microscopy (generally analyzing the images after the experiment and after applying different filters to the data to make things stand out), after which another experiment is performed. If biological features that are sought after could be analyzed and detected in real time, the experiments could become more modular, allowing a researcher to apply a chain of experiments together, saving time and money.

This is very complex feature analysis that would have extremely accurate though. So the bare bones fundamental structure, such as performing lineage tracking and extracting features, would have to be able to function well.

Features: Use Case/Scenario

Our primary use case is to be able to detect and tag the cells in each frame in a video. That is, given a frame determine what type of cell and its location in the image. Counting the total number of cells is also biologically useful and an extension of this case.

Our current implementation is to use SIFT features to determine important key points for a yeast cell and identify those points within the video for each frame. Unfortunately, this is not working well for our data, even to the point where it cannot find the image which we trained on for determining the SIFT points of a yeast cell within the video. This could be due to how few SIFT points we found within the image, but changing the amount of points we generate on the image and video still hasn't improved our methods of detection.

Training on more images of yeast and aggregating those features through image analysis aggregation schemes like Fischer Vector, VLAD, or Super Vector may yield much better results than just training and trying to detect one image. Using these schemes in MATLAB is an easy task but fairly uncertain one for Java.

Tagging what state the cell is in is also important and a second use case. We wish to be able to tag a yeast cell that is budding, that is there is a smaller yeast cell growing conjoined to it, or E.Coli splitting, where E.Coli grows long and then tapers in the middle before it splits entirely into two different sister E.Coli.

Here we are having a better job of identifying yeast cells that are budding, at least for the image we trained on, however even though SIFT is rotation invariant, it wasn't able to pick up budding cells that are in a different orientation. Again, training on more than one image and aggregating our features through an image analysis aggregation schema could improve our detection powers for classifying whether an image contains budding or splitting cells and annotate them correctly.

These two use cases would allow us to map states of one image to another to determine lineage and map a lineage tree the cells. This is highly adventurous though and likely will not be finished within the project's semester scope.

Approach

- Data Sources
    - Published cellular research with microscopy movies (frames taken every 2-5 minutes). Youtube also offers video data of microscopy as well, generally with frames taken within a shorter window, but for lesser duration. The research data also tries to analyze more complicated mechanisms while the youtube data displays features such as budding and reproduction.
    - Research papers that we have looked at may be found in our literature review, and an example of youtube data would be the following:
      https://www.youtube.com/watch?v=iOvrq6ssy2Y
      https://www.youtube.com/watch?v=BTHYaf-EuYs
    - We may also have access to a usb microscope allowing for real time streamed microscopy. The microscope is 500x which would allow viewing of yeast cells on a basic level.

- Analytic Tools
    - Spark- Apache Spark allows us to perform data parallelism through the mapreduce paradigm, allowing us to train and learn from our data in order to build a model to classify what parts of our images are cells and what state they exist in.
    - OpenImageJ- OpenImageJ is a library that allows the user to extract key points and frames through SIFT point detection and tagging.

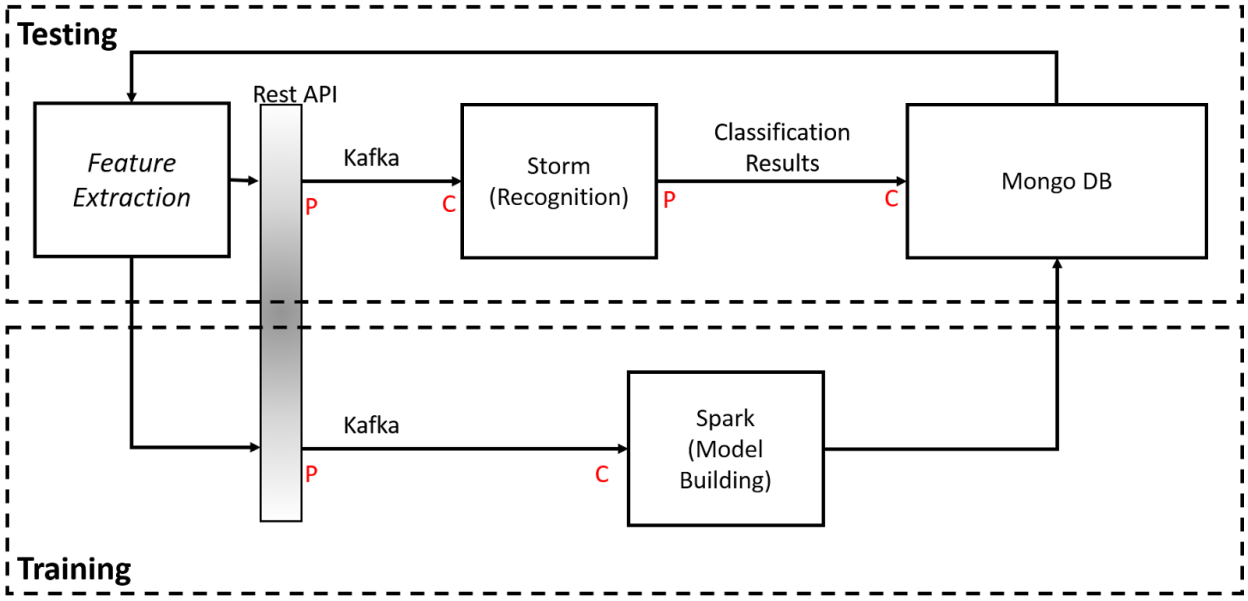
    - Spark MLLib- the Spark MLLib is a rich resource of machine learning, allowing us easy access to decision trees, random forests, feature analysis, and neural network schemes.
    - Storm- Storm allows us to perform task parallelism so that when we stream in videos as frames, we are able to perform many tasks on those frames in order to determine where cells are and what state they are in.
    - Kafka- Kafka is a publisher subscriber system that allows a robust high volume queuing of data from which Spark and Storm can act as both consumers and producers of different topics in order to communicate between themselves and the client.
- Analytical Tasks
    - Track and detect cells from one frame to another
        - Requires ability to detect individual cells
        - Requires ability to differentiate or map cells from one frame to another
    - Classify and annotate whether a cell is dividing or budding
    - Train on multiple images and aggregate SIFT features from them
    - Extract many features from cells.
        - Note: This could be physiological such as size and shape or behavior such as splitting and interactions among cells, or triggered, such as turning on a fluorescent light to see a gene expression tag glow.
    - Determining if slide needs to be adjusted relative to microscope
    - Counting number of cells in view
- Expected Inputs/Outputs
    - The expected input would be a video of cell microscopy where we output a lineage tree or video as well as biological tagging (such as knowing when it's splitting) or what gene or protein it contains via fluorescent tagging.
    - Another input could be a video stream from a microscope where each frame is processed.
    - These inputs would be processed and serve as inputs for other functions within the program as well. Such as processing the video to make SIFT points easier and more meaningful to extract.
    - Outputs would be a lineage tree of cells, ability to find locations within the video of biological features such as splitting and budding, and counts of the number of cells seen in total or with a certain feature.
    - Other outputs: Metadata on video such as attribute measurements at a specific time or average values over length of video
- Algorithms
    - SIFT aggregation schemes: Training on one image and using it as representative of an entire class doesn't work and really doesn't work with our data. As such, we need to aggregate many sift points from different cells of the same type and group them to classify what is a yeast cell, what is a budding cell, what is an E.Coli, etc. These methods include Bag of Words at its most primitive, evolving into VLAD, Fischer Vector, and Super Vector as more complicated schemes.

    - Random Forest/Decision Tree algorithms
    - Object detection, aggregation, and segregation using image analysis.
        - Note: The current key point extraction technique using SIFT as shown in Lab2,3 does not accurately find single E.Coli in our videos. E.Coli are all fairly regular and contain a lot of the same curvature (that SIFT tries to find and map as a key point). E.Coli also grow and wiggle changing their shape, making curvature based methods difficult for tracking it. We also have yeast cell data tend to have more distinctive features within them and around them, which could be easier to detect and track.

Related Work

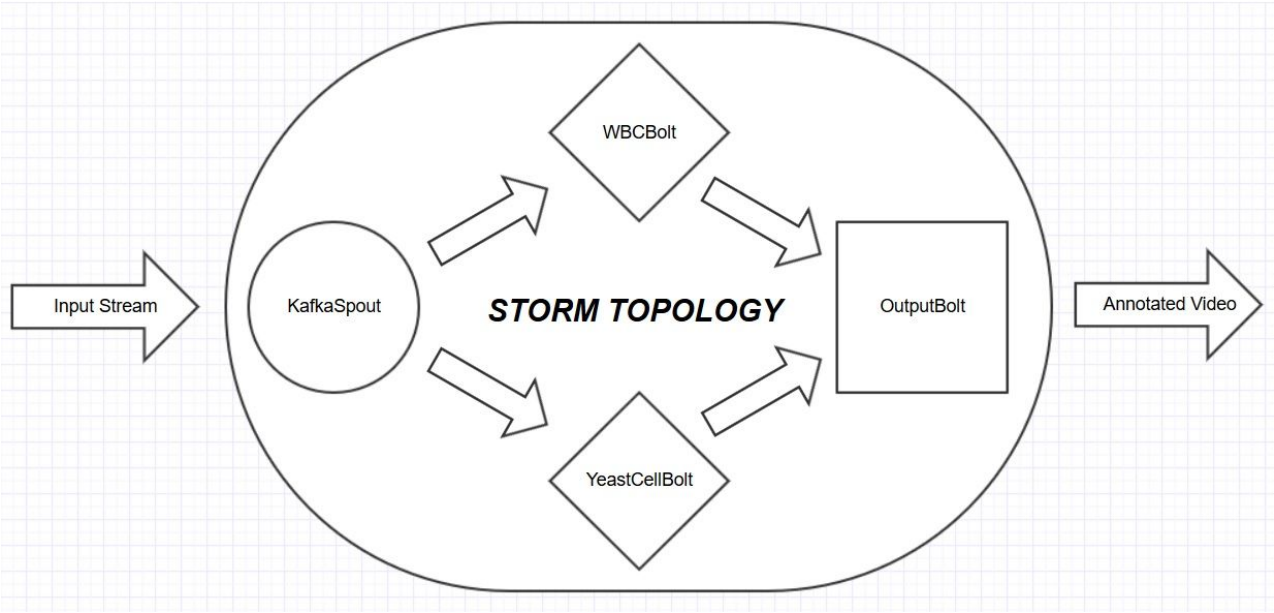
- Open Source Projects
    - MicrobeJ for ImageJ (I don't think MicrobeJ is opens source though, but ImageJ is)
- Literature Reviews (some are from proposal)
    - http://www.ncbi.nlm.nih.gov/pmc/articles/PMC4820279/ (ecoli videos)
    - http://journals.plos.org/plosone/article?id=10.1371/journal.pone.0032621 (yeast)
    - http://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1003396#s5 (yeast)
    - http://www.ncbi.nlm.nih.gov/pubmed/27572972
    - Chen, Zhou & Wong. "Automated segmentation, classification, and tracking of cancer cell nuclei in time-lapse microscopy." IEEE Transactions on Biomedical Engineering, vol. 53, no. 4, April 2006. http://ieeexplore.ieee.org/document/1608529/
    - Dewan, Ahmad & Swamy. "A Method for Automatic Segmentation of Nuclei in Phase-Contrast Images Based on Intensity, Convexity and Texture." IEEE Transactions on Biomedical Circuits and Systems, vol. 8, no. 5, October 2014. http://ieeexplore.ieee.org/document/6762958/
    - Goutam & Sailaja. "Classification of acute myelogenous leukemia in blood microscopic images using supervised classifier." 2015 IEEE International Conference on Engineering and Technology (ICETECH), March 2015. http://ieeexplore.ieee.org/document/7275021/
    - Tran, Pham & Zhou. "Cell phase identification using fuzzy Gaussian mixture models." Proceedings of 2005 International Symposium on Intelligent Signal Processing and Communication Systems, 2005. http://ieeexplore.ieee.org/document/1595447/
- Application Specification
    - OpenImaj/SIFT
- System Specification/Software Architecture
    - Spark - ML to assign probabilities of specific features in a frame
    - Storm - efficiently handle feature extraction from cell video data
    - Kafka - transferring video data between systems

    - Potentially utilizing MongoDB to store data that can be used to train and build our model for cell detection
    - AWS cloud - cloud computing platform we will utilize to implement the above technologies in real time
- Design of Big Data Analytics Server
    - Parallelism/Distribution: Task/Data
        - Data parallelism of splitting image into frame then applying ML algorithms to train for object classification.
        - Task parallelism within a frame to determine edges/sift points/biological tagging through color analysis or shape analysis.
    - General Workflow (as taken from lecture):



        - Training: Data is extracted and sent to Spark (via Rest API). Spark builds a model and sends the corresponding If-Else decision tree to MongoDB
        - Testing: Data is extracted and sent to Storm (via Rest API). Storm classifies cells within video, draws a box around classified cells and labels based on the chosen classification. The annotated video is stored in MongoDB, and can be retrieved by the client.

    - Storm Topology:



- KafkaSpout: Consumes streaming data from client
- WBCBolt: Bolt that determines (based on generated If-Else decision tree) if incoming feature data is a White Blood cell. If so, it identifies the cell and labels it
- YeastCellBolt: Bolt that determines (based on generated If-Else decision tree) if incoming feature data is a Yeast cell. If so, it identifies the cell and labels it.
- OutputBolt: Combines data in the form of an annotated video and outputs video to MongoDB

- Design of Mobile Client (smartphone/web): N/A
- Existing Applications/Services Used: Time-lapse microscopy videos (http://www.ncbi.nlm.nih.gov/pmc/articles/PMC4820279/)

Implementation

- Local client application to extract key feature data
- Send training data to SparkModelBuilder, which sends output of Classification Model in the form of an If-Else decision tree to MongoDB
- Storm topology is generated based on model in MongoDB
- Created topology is uploaded and run on storm server
- Test data from client is sent to storm and is classified based on cell type (i.e. blood cells or yeast cells).
- Output of classification is sent to MongoDB, which includes video that boxes cell type with label.
- Video can be retrieved for the client from MongoDB via REST call.

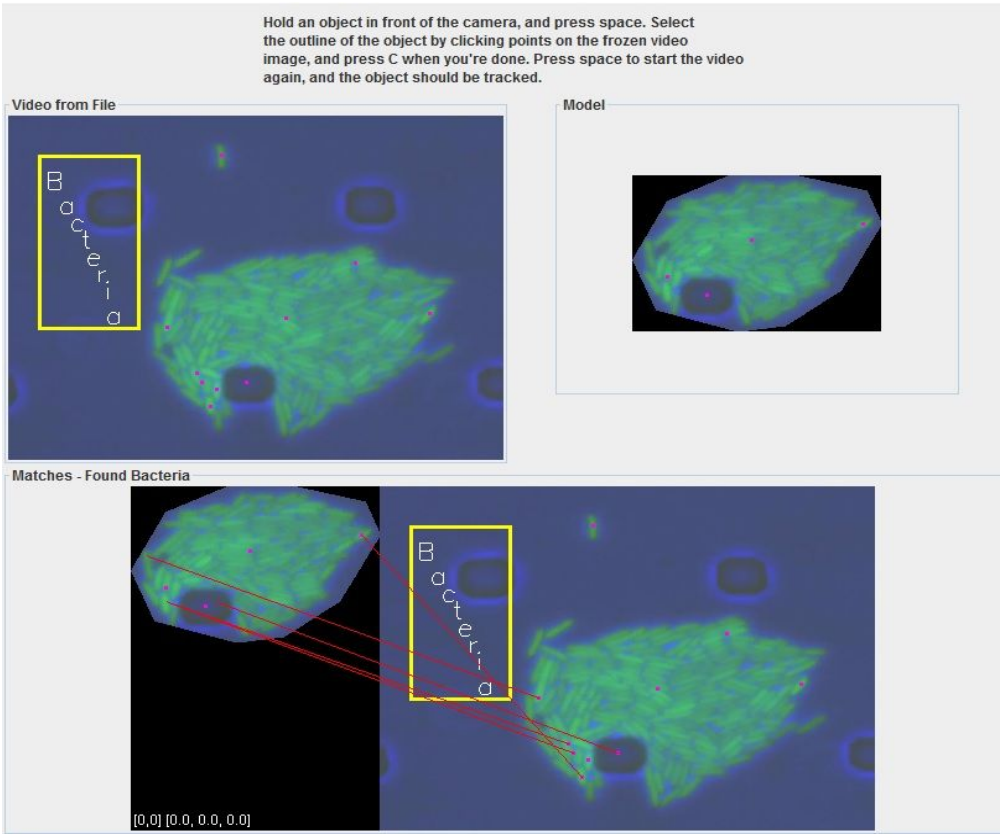
Documentation

Proposal
Source: https://github.com/DRinKC/RT-BigData-Project_Team1/tree/master/Proposal

Lab3
Source: https://github.com/DRinKC/RT-BigData-Project_Team1/wiki/Lab-3
Metadata output from Main Frame Detection:

```
The original image had 141 frames and took 6097 milli seconds and our new video had 30 frames and took 857 milli seconds
```

Lab4
Source: https://github.com/DRinKC/RT-BigData-Project_Team1/wiki/Lab-4 \
Object Detection & Tracking with annotation:



Lab5
Source: https://github.com/DRinKC/RT-BigData-Project_Team1/wiki/Lab-5
Object Detection and Annotation:

Decision Tree output:

```
DECISION TREE RESULTS

CONFUSION MATRIX:
1.0  0.0  0.0
0.0  1.0  0.0
0.0  0.0  1.0
F-MEASURE: 1.0
RECALL: 1.0
PRECISION: 1.0
```

Random Forest output:

```
RANDOM FOREST RESULTS

CONFUSION MATRIX:
1.0  0.0  0.0
0.0  1.0  0.0
0.0  0.0  1.0
F-MEASURE: 1.0
RECALL: 1.0
PRECISION: 1.0
```

Lab6

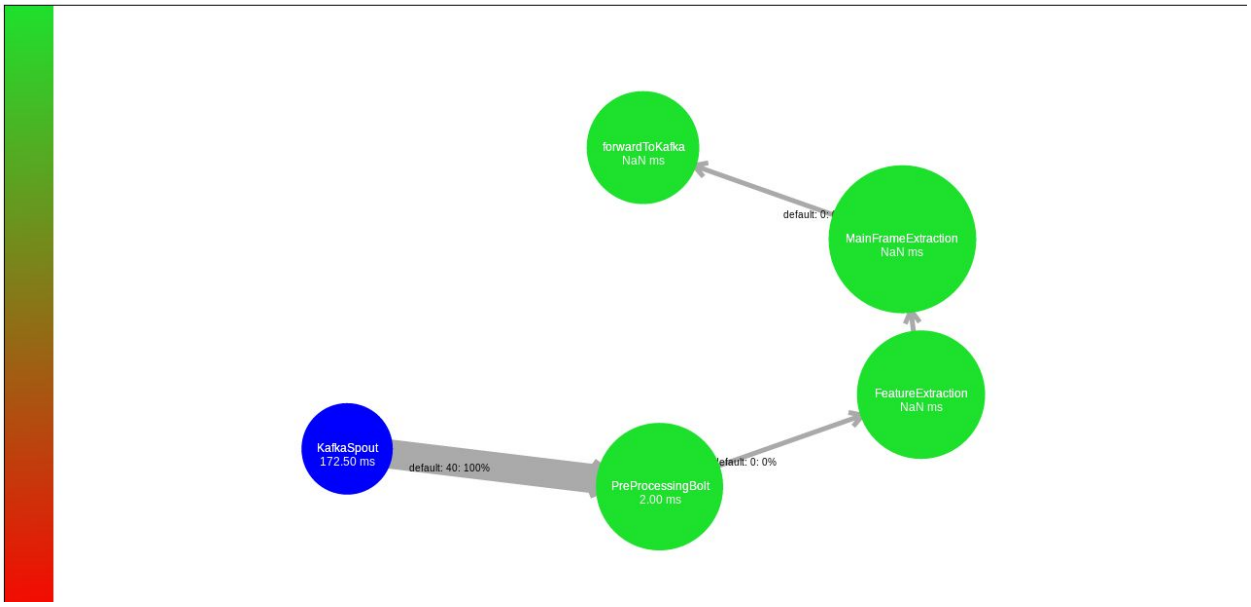
Source: https://github.com/DRinKC/RT-BigData-Project_Team1/wiki/Lab-6

Extracting key frames, metadata and features, and sending data to Storm via Kafka Producer (no screenshot available)

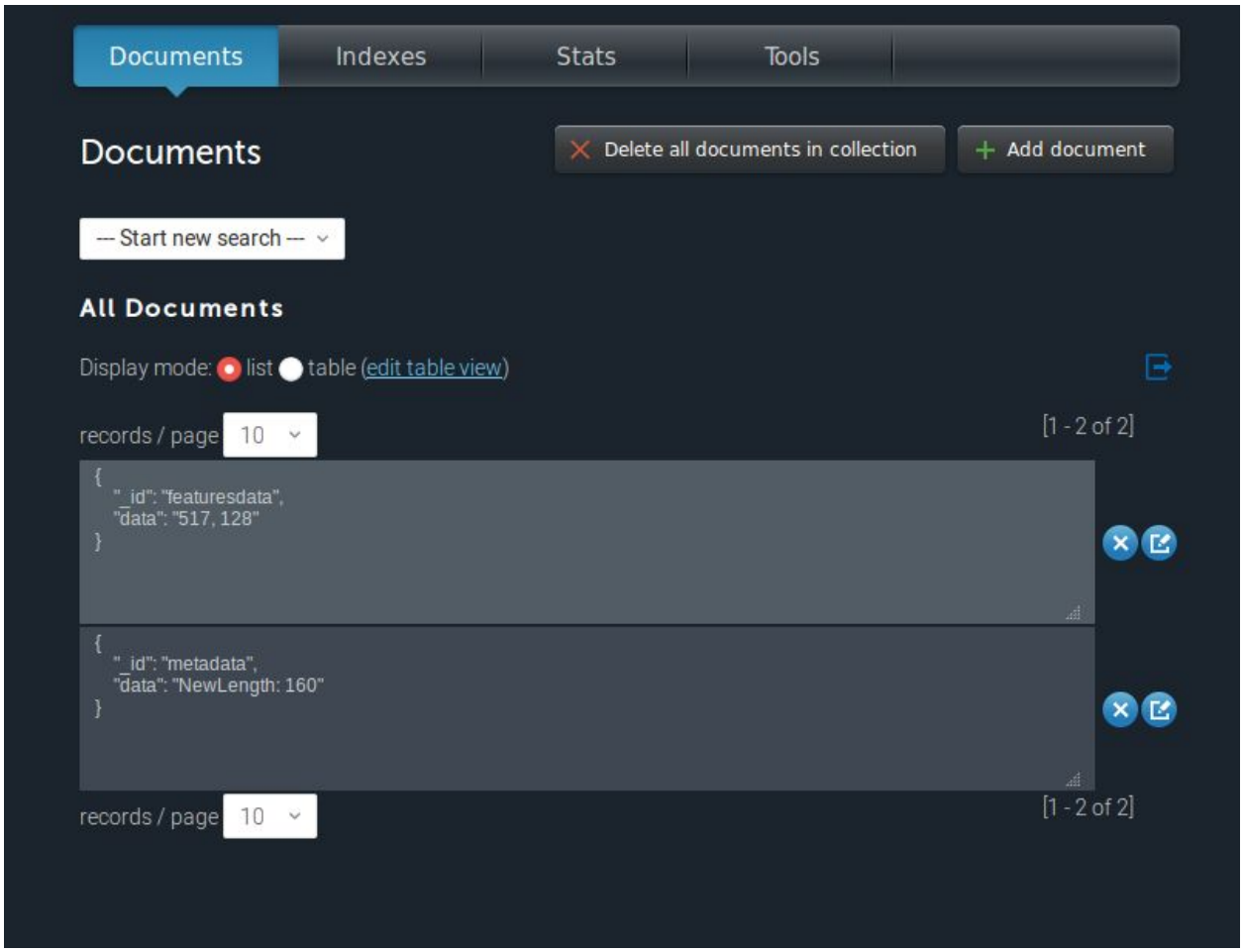
Lab7-8

Source: https://github.com/DRinKC/RT-BigData-Project_Team1/wiki/Lab-7,8

Creating Storm topology to analyze extracted information



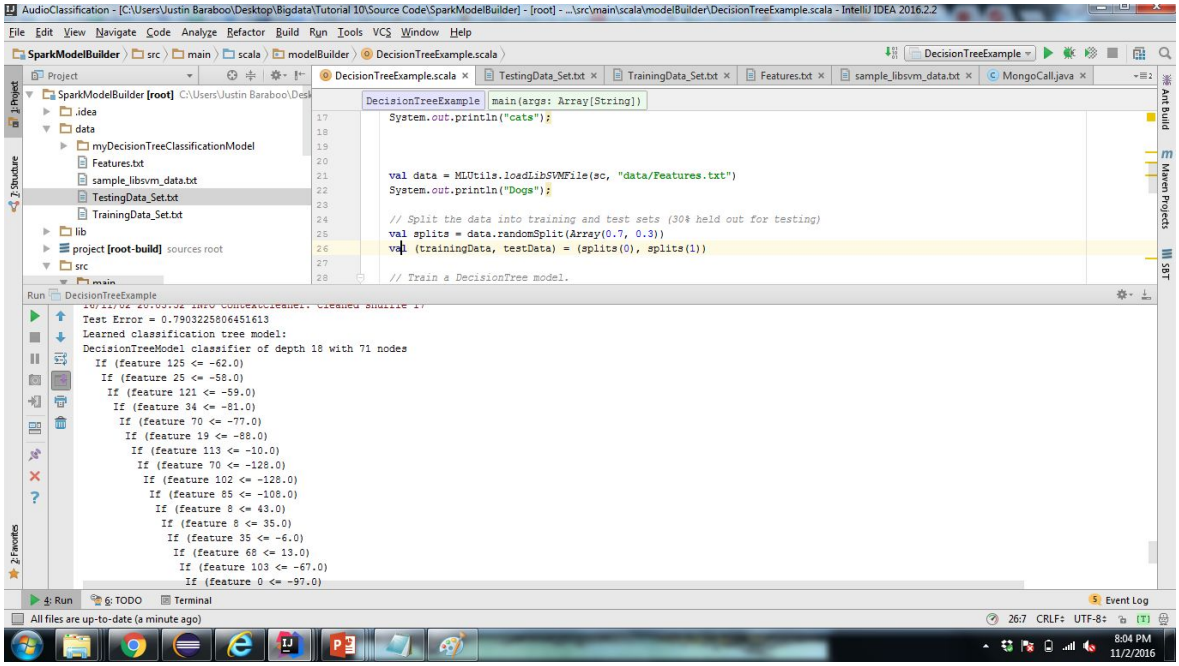
Sending results to MongoDB



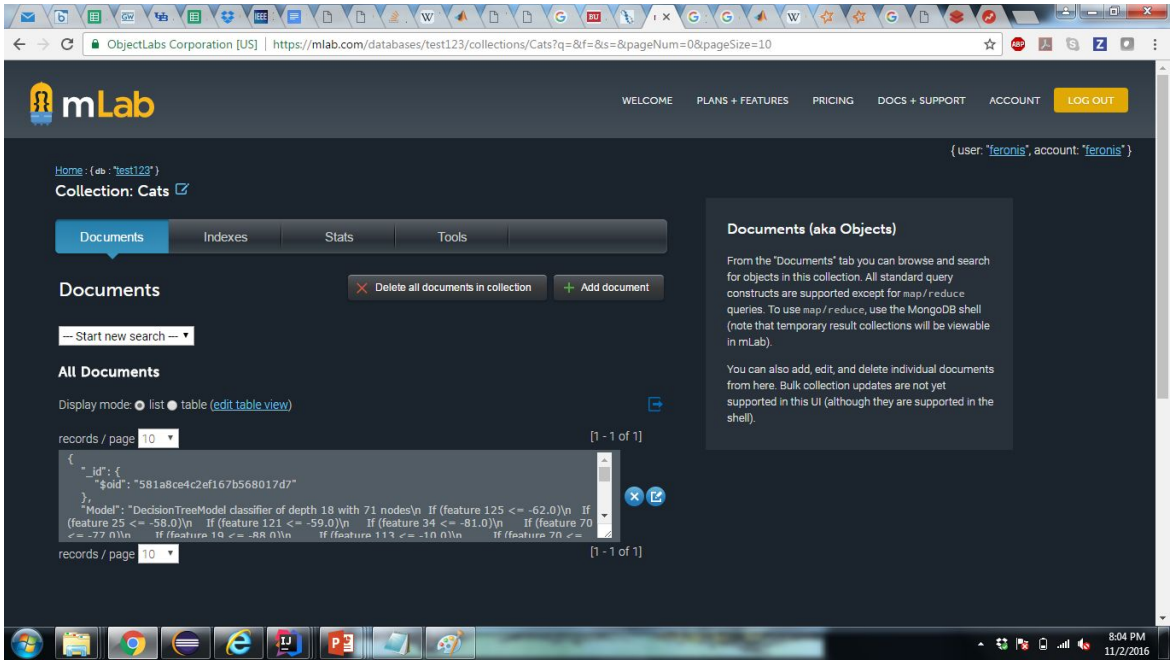
Lab9

Source: https://github.com/DRinKC/RT-BigData-Project_Team1/wiki/Lab-9

Creating If-Else decision tree with SparkModelBuilder



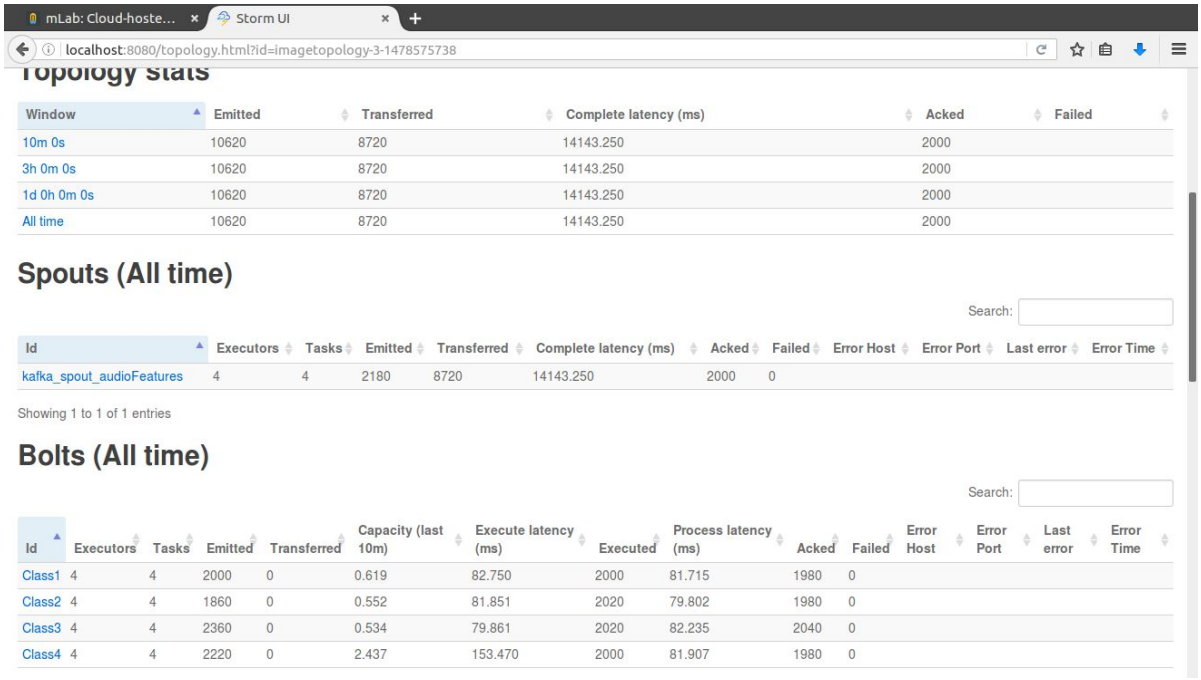
Sending If-Else decision tree to MongoDB



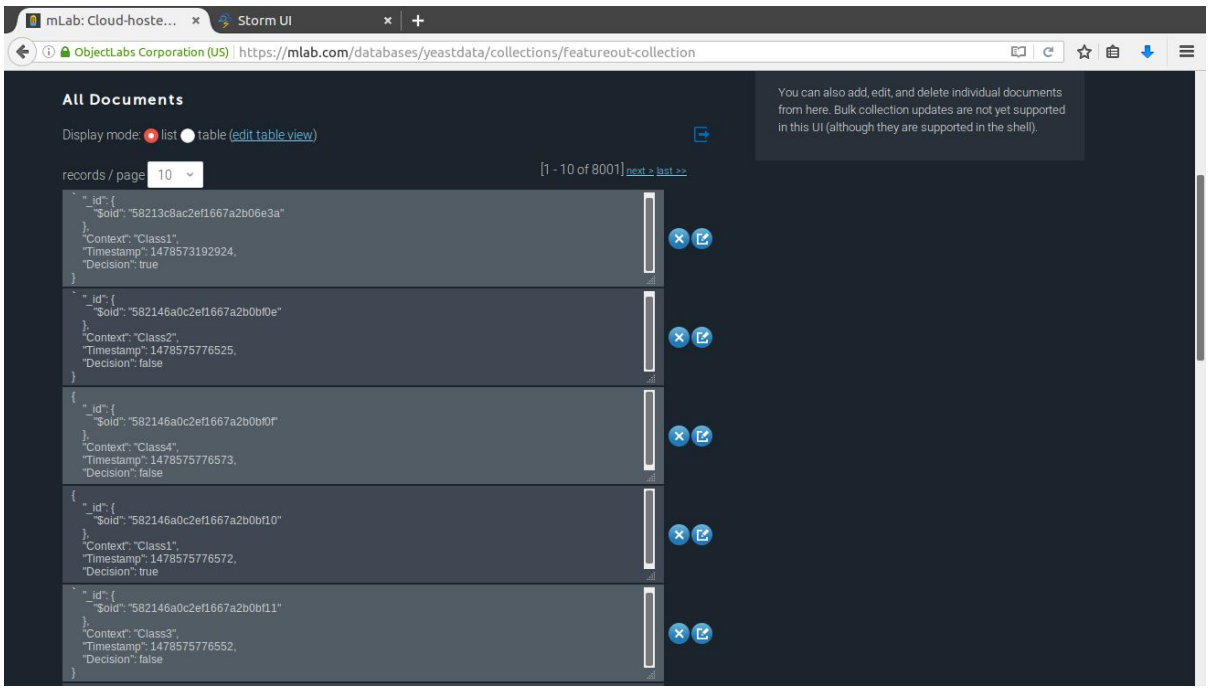
Lab10

Source: https://github.com/DRinKC/RT-BigData-Project_Team1/wiki/Lab-10

Generating Storm topology based on decision from MongoDB



Sending data to be classified through Storm whose output is sent to MongoDB



Project Management: Implementation status report

- Work completed:
    - Description:
        - Creating Storm Bolts
        - Testing workflow of Storm without Output stored in MongoDB

        - Generating If/Else model and storing decision tree in MongoDB
        - Generating new Topology based on decision tree,
        - Testing new Storm topology
        - Researching USB-Microscope options/streaming data
    - Responsibility:
        - Bill: Researching Streaming Data
        - David: Creating Storm bolts; assisted with testing Storm workflow
        - Brendan: Testing Storm workflow, generating new topology and testing new topology; assisted with creating Storm bolts
        - Justin: Generating If/Else model and storing decision tree in MongoDB
    - Contributions (members/percentage): roughly equal (25%)
    - Work to be completed
        - Description:
            - Gathering test and training data
            - Rest API to retrieve annotated video
            - Improved classification of cells
            - Upgrade to Heron
            - Analyzing results (e.g., creating Confusion Matrix)
        - Responsibility:
            - Bill: Gathering test and training data
            - Brendan: Upgrading to Heron, Rest API to retrieve video from Mongo
            - David: Improving classification of cells, Rest API to retrieve video from Mongo
            - Justin: Improving classification of cells, analyzing results
        - Time to be taken (estimated #hours):
            - Bill's task: Data collection (3-5 hours)
            - Brendan's task: Heron (2 hours), API (2-3 hours)
            - David's task: Classification (2-3 hours), API (1-2 hours)
            - Justin's task: Classification (2-3 hours), Analysis (1-2 hours)
- Issues/Concerns:
    - Amount of training/test data seems sparse. Mostly we will have to rely on videos from YouTube, which means we have very little control on the variance of cell types
    - Model may not be accurate given that test and training data will be sparse and variable.
    - Implementing new system/upgrade (Heron) could lead to unforeseen issues that may be difficult to troubleshoot and prevent progress on improving overall project's design and accuracy.
    - Getting entire workflow operating at the same time
        - Server reliability will likely make it hard to have much time to test entire workflow