Team 1

Crawling For Breach Reports

Team Members: Alec Lones, Jeremiah Brusegaard, Mark Schwartz, Nolan Kim
Faculty Advisor/Client: Benjamin Blakely
Team Website: http://sddec19-01.sd.ece.iastate.edu/

Problem Statement

Currently, there is no good way to be notified about every data breach that occurs, so important data breaches can easily fly under the radar. This can be dangerous for a company's security team, who needs to stay up to date with the latest data breaches in order to keep their company secure. In addition, it is important for these data breaches to be stored for security teams to reference later.

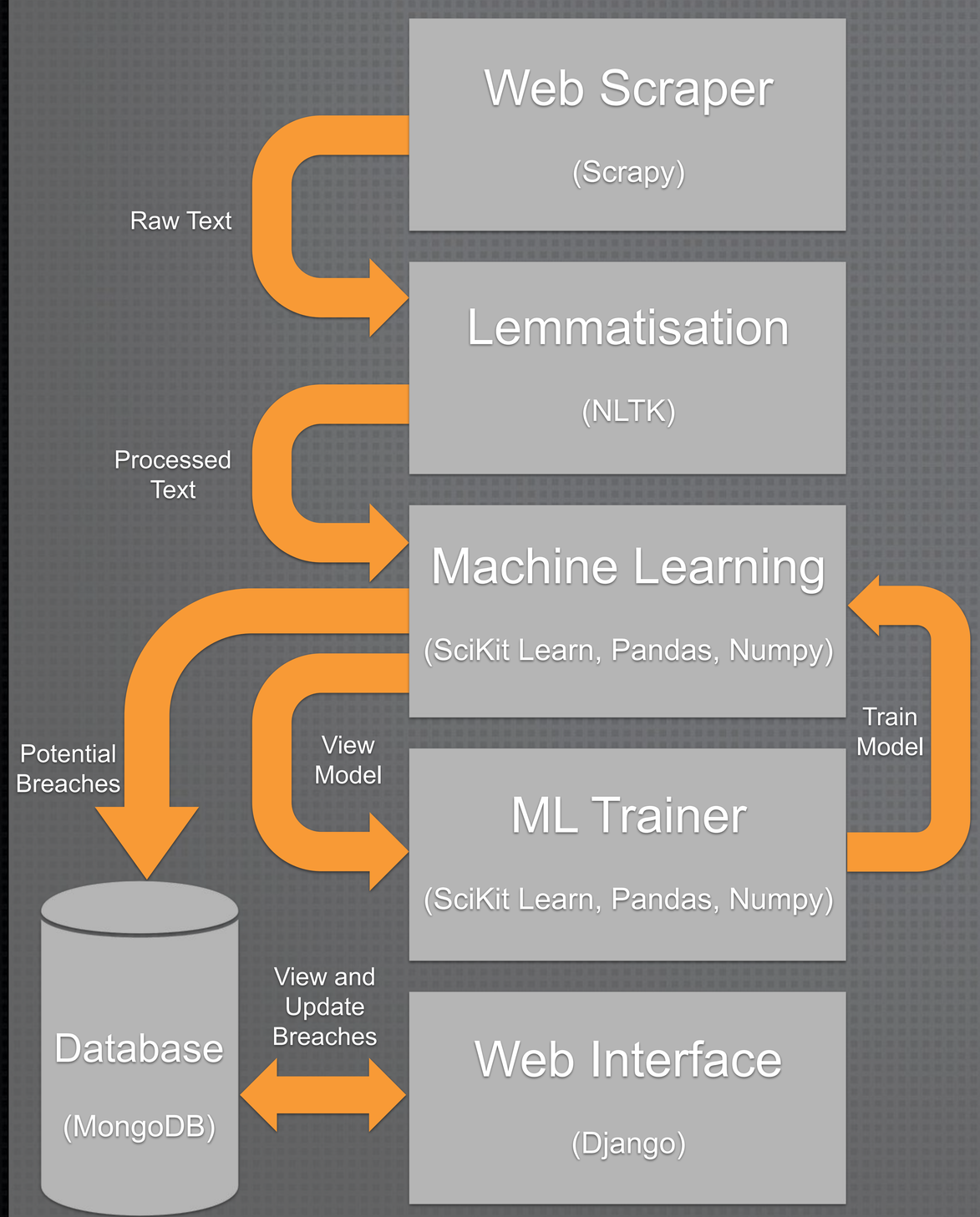
Solution

This project will serve as an early warning for Chief Security Officers on data breach reports that may affect their company. We accomplished this by implementing a web scraper to traverse the internet and identify data breach reports using machine learning. Our scraper then stores the breach reports in a database for future reference. Finally, an analyst will be able to validate and select relevant breaches through a web interface. With this information the Chief Security Officer will be informed on current security threats to their organization.

Design Approach

Our design is fairly straightforward. Our web scraper scrapes the web to pull down webpage text, which gets lemmatized to make it easier for the machine learning module to categorize. Our machine learning module decides whether or not the page is a breach report. At this point in the process, a human worker can correct the algorithm to train the model. Data breach reports are then stored in a database, and can be accessed via a web interface.

Block Diagram



Functional Requirements

- System must scrape the internet
  - Data will only be pulled from the internet
- System must save pages that contain data breach reports
  - Pages flagged as data breach reports will be saved into the database
  - Pages verified by an analyst will be marked as so in the database
- System must allow for training of the Machine Learning model
  - System comes with a model but allows training of a new one
- System must support displaying of aggregate data
  - Web interface will display data and allow analyst to verify results

Non-Functional Requirements

- Python will be the primary programming language
- Breach reports are limited to the English language
- Crawler will crawl the internet at a speed slow enough speed to not get blacklisted by sites

Operating Environment

- Linux Operating System
  - Other OSes are also supported, but default model will not integrate
- Recommended 16 GB RAM or more
- Recommended 4 CPU cores or more

Technical Details

- Programming Language: Python 3
- Libraries
  - Web Scraper Scrapy
  - Lemmatizer - NLTK (Natural Language ToolKit)
  - Machine Learning Module - SciKit Learn, Pandas, Numpy
  - Database - MongoDB
  - Web Interface - Django

Development

- Tools and Environment
  - Pycharm
  - Ubuntu 19.10 Server (Hosting Application)
  - VirtualBox

ML Model

- Random Forest Classifier
- Training Data Results:
  - False Positive Rate: 0.005%
  - False Negative Rate: 0.054%
  - Matthews Correlation Coefficient: 0.95

Most Influential Words

Word	Importance Value
breach	0.073
security	0.063
attack	0.055
hack	0.039
malware	0.026
cybersecurity	0.025
hacker	0.021
victim	0.019
compromise	0.017

Unit Testing

- Software is broken into 5 main modules: the scraper, lemmatizer, vectorizer, machine learning, and database
- Used PyUnit to unit test each individual module as much as possible with the machine learning module being the exception

ML Model Testing

- Used Cross validation with random subset sampling
- Used Matthews correlation coefficient for validating our true positive, true negative ratios supplemented with the confusion matrix

Note: Training data set was supplied by the client

Most Relevant Standards

- PEP 8 Code formatting
- IEEE 1008-1987 - Unit Testing standards
- IEEE 829 - Documentation Standards