Background Checks

By: Michael Kulinich, Steven Abouchedid,
Erik Hombledal , Corey Spielman

1. Initial Discussions

Brainstorming and Potential Solutions

Initial Approach

- 2 Possible Paths
  - HTML Scraping or Optical Character Recognition
- Decided to Pursue Both
  - BeautifulSoup4(HTML) and Tesseract(OCR)
- Both Require Selenium

Tesseract vs. BeautifulSoup

Tesseract

- Versatile
  - *Requires Screenshots*
  - *Website Layout Inconsistency*

BeautifulSoup4

- Lightweight
- Straightforward application
- *Does not work with JS websites*
  - *Inconsistency in HTML Tags*

What We Decided

- Tesseract + Selenium
  - Works on websites BeautifulSoup4 doesn't
  - Selenium required for navigation + functionality

Tesseract OCR

2. Our Concept

pyTesseract OCR + Selenium

Why Selenium

- Navigating websites
  - Name input into fields, confirmations
- Screenshots through Selenium
  - Taken by scrolling down page through Selenium, and saving to designated folder

How Tesseract Works

- Pre-trained Deep Learning algorithm
- Takes screenshots as input, then outputs to text
  - High accuracy, but intensive

“Close Enough” Philosophy

- Impossible to account for every variation
- Rule out possibilities
  - Ex. A case number will never have a space

DEMO

Our “close enough” design philosophy is easier to show

3. Moving Forward

Plans for Improvement

Our Process in Theory

Initial Name Scan

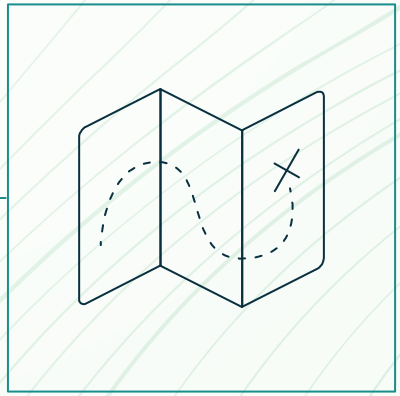
- Given name, returns basic output, if any:
- EX.
  - Case ID
  - Jurisdiction
  - Class of Crime
  - Ruling
  - Year

Full Scan (Optional)

- Given case ID, return a full report
- EX.
  - Sentencing
  - Fines
  - Plaintiff / Def.
  - Probation

Decision

- Can set criteria for automatic pass/fail
- Full scan used for manual review



Thanks!

Any questions?