18734 Homework 3

Released: October 7, 2016
Due: 12 noon Eastern, 9am Pacific, Oct 21, 2016

1 Legalease [20 points]

In this question, you will convert a simple policy into the Legalease language. To review the Legalease language and its grammar, refer to the lecture slides from September 28. You may find it useful to go through a simple example provided in Section III.C in the paper *Bootstrapping Privacy Compliance in Big Data Systems* [1].

A credit company CreditX has the following information about its customers:

- Name
- Address
- PhoneNumber
- DateOfBirth
- SSN

All this information is organized under a common table: AccountInfo. Their privacy policy states the following:

> We will not share your SSN with any third party vendors, or use it for any form of advertising. Your phone number will not be used for any purpose other than notifying you about inconsistencies in your account. Your address will not be used, except by the Legal Team for audit purposes.

Convert this statement into Legalease. You may use the attributes: *DataType, UseForPurpose, AccessByRole*. Relevant attribute-values for *DataType* are *AccountInfo, Name, Address, PhoneNumber, DateOfBirth,* and *SSN*. Useful attribute-values for *UseForPurpose* are *ThirdPartySharing, Advertising, Notification* and *Audit*. One attribute-value for *AccessByRole* is *LegalTeam*.

[1] `http://www.andrew.cmu.edu/user/danupam/sen-guha-datta-oakland14.pdf`

2 AdFisher [35=15+20 points]

For this part of the homework, you will work with AdFisher. AdFisher is a tool written in Python to automate browser based experiments and run machine learning based statistical analyses on the collected data.

2.1 Installation

AdFisher was developed around 2014 on MacOS.

However, due to a recent change in the web driver used by Firefox, AdFisher does not work with the most recent versions of Firefox. We have tested and found that AdFisher works with Version 45 of Firefox. Past versions of Firefox can be downloaded from the Mozilla release archives [2].

Because AdFisher is not currently compatible the most recent version of Firefox, and because Firefox automatically updates itself on MacOS, non-linux users should install a VM software like VirtualBox [3] to run AdFisher on it.

In order to get Version 45 of Firefox on Ubuntu [4]:

```
sudo apt-get remove firefox
wget https://ftp.mozilla.org/pub/firefox/releases/45.0/linux-x86_64/en-US/firefox-45.0.tar.bz2
tar -xjf firefox-45.0.tar.bz2
sudo mv firefox /opt/firefox45
sudo ln -s /opt/firefox45/firefox /usr/bin/firefox
```

Running `firefox --version` should return: `'Mozilla Firefox 45.0'`.

Clone the git repository for AdFisher by running the following command on your terminal:

```
git clone https://github.com/tadatitam/info-flow-experiments.git -b 18734
```

In your current directory, you will find the repository folder `info-flow-experiments`. Follow the instructions on `https://github.com/tadatitam/info-flow-experiments` to install package dependencies for AdFisher. Go to `AdFisher/examples/` and run `demo_exp.py` and `demo_analysis.py` to check if all dependencies have been correctly installed. For this homework, you do not have to install xvfb or xvfbwrapper as long as you do not intend to run experiments in headless mode (i.e. keep `headless='False'` in the definition of `make_unit()`).

You may use the `possibility.cylab.cmu.edu` server you used for Homework 1. However, you have to ssh into the server and run the experiments in headless mode, which means you will not be able to see the browsers while the experiments run. For the headless mode, you have to install xvfb and xvfbwrapper, and set `headless=True` on line 14 of `demo_exp.py`. We also noticed that before the experiment is complete, ssh sessions expire, which terminates the running experiment. So, you have to start the experiments within a screen session [5].

If you want to run experiments in headless mode on your Ubuntu computer, you can follow the above instructions. However, you do not have to use screen. Note that, Mac OS does not have any xvfb package alternative, so you cannot run experiments headless on your Mac.

[2] `https://ftp.mozilla.org/pub/firefox/releases/`
[3] `https://www.virtualbox.org/wiki/Downloads`
[4] `http://stackoverflow.com/questions/37761668/cant-open-browser-with-selenium-after-firefox-update`
[5] `http://aperiodic.net/screen/quick_reference`

2.2 Test scripts [15 = 2+2+2+9]

In the `examples` folder, you will find the script `test.substance.py`. This is a toned down version of the original experiment we ran [6]. Follow the README on the git page to answer the following questions about the `test.substance.py` script:

2.2.1 **[2]** In order to compare two treatments, AdFisher launches two groups of browser agents and applies one treatment to all agents in one group, and the other treatment to all browsers in the second group. Treatments are essentially online activities that the browsers perform. What two treatments are being compared in `test.substance.py`? In particular, state the sequence of actions that browser agents in each group perform.

2.2.2 **[2]** After performing actions, all agents collect measurements from the web. AdFisher can collect demographics inferred in Google Ad Settings [7] or ads from some news websites. Which site are the browser agents collecting ads from? How many times is the collection site being reloaded on each agent for ad collection? What is the delay between successive reloads of the collection site?

2.2.3 **[2]** In order to run experiments at scale, AdFisher runs experiments in several blocks. In each block, a number of browser agents collect data. For how many blocks is `test.substance.py` set to run? How many agents (browser instances) are being launched in each block?

2.2.4 **[9]** Run the `test.substance.py` script by executing the following command:

- python test.substance.py

If the experiment is running properly, you should see something like the following on your terminal:

```
Beginning Collection.
Ran 1 test in 5.834s
OK
Collection Complete. Results stored in substance.txt
Starting Experiment
Block 1
....................
Ran 1 test in 170.295s
OK
Instance 1 exiting!
..
Ran 1 test in 174.639s
OK
Instance 0 exiting!
```

... and so on. The experiment should take about 90 minutes to complete. At the end of the collection, AdFisher carries out analysis on the collected data. For the analysis, AdFisher splits the

[6]`http://arxiv.org/abs/1408.6491`

[7]`https://www.google.com/settings/ads`

collected data into the training and testing data. It applies machine learning on the training data to learn differences in the ad distributions to the two groups. It then tests whether the differences are significant on the testing data. The results from the analyses are printed out on the terminal. Among the results, is the p-value, which is a measure of significance for the results. The p-value is significant when its value is less than 0.05. A significant p-value indicates that the ad distributions to the two groups are significantly different

For the data collected in `test.substance.py`, the analysis takes about 40 seconds to run after collection is complete. You do not have to run any additional commands for the analysis. The script `test.substance.py` is written so that analysis is carried out automatically after collection. Use the displayed results to answer the following questions: What is the test-accuracy? What is the resulting p-value? Is the p-value significant?

2.3 Your Script [20=2+7+7+4]

In this subpart, you will design an experiment and add code to the `hw3.py` script to run experiments with AdFisher. Note that in the script, a tab is replaced by 4 spaces. So, in case you get an Python indentation error, use 4 spaces instead of tabs.

Scenario: Alice is worried that her online activities are being used by Google target recommendations to her. She uses her browser to visit websites, perform Google searches and reads news from Google News. She is worried some or all of these activities are affecting the ads and/or news articles that are served to her. She hires you, a privacy technologist, to address her concerns. Your job is to design and run an experiment using AdFisher to address her concerns.

You have the following functions available to you to design and run experiments:

- Treatments
  - `unit.visit_sites(site_file, delay)`: This function is written in `browser_unit.py`. `site_file` is the path to a file which contains a list of websites in each new line. This method drives the browser to visit all these websites. Example use: `unit.visit_sites (site_file = ‘substance.txt’, delay = 5)`.
  - `unit.search_and_click(query_file, clickdelay, clickcount)`: This function is written in `google_search.py`. `query_file` is the path to a file which contains a list of queries in each new line. This method drives the browser to search for a query term, click on `clickcount` search results, with `clickdelay` seconds of delay between subsequent clicks. Example use: `unit.search_and_click(query_file = ‘queries.txt’, clickdelay = 20, clickcount = 5)`.
  - `unit.read_articles(count, agency, keyword, category, time_on_site)`: This function is written in `google_news.py`. This method loads the Google News page and finds articles based on certain search criteria - the news `agency` that serves the ad, presence of a `keyword` in the snippet of the article, or the `category` of news. An exact string match is performed to find corresponding news articles. The browser clicks on each of these news articles, spending `time_on_site` seconds on the site before moving on to the next article. NOTE: If you do not provide enough delay between clicks, Google may block you. `count` specifies an upper bound on the number of articles visited. Example use: `unit.read_articles(count=5, agency=‘CNN’, keyword=None,`

`category='Business', time_on_site=10)` visits at most 5 articles served by 'CNN' or in the category 'Business', spending 10 seconds reading each article.

- Measurements
  - `unit.get_news(type, reloads, delay)`: This function is written in `google_news.py`. This method collects news articles from Google News by reloading the page `reloads` times, with `delay` seconds between successive reloads. `type` can be either 'top' to collect only the Top News, or 'all' to collect all articles on the page. Example use: `unit.get_news(type='top', reloads=5, delay=10)`.
  - `unit.collect_ads(reloads, delay, site)`: This function is written in `google_ads.py`. This method collects Google text ads served on the Times of India ('toi') or BBC ('bbc') by reloading the page `reloads` times, with `delay` seconds between successive reloads. Example use: `unit.collect_ads(reloads=10, delay=5, site='bbc')`.

Note that AdFisher currently cannot analyze two types of measurements at the same time. If your design includes collection of both news and ads, you need to run two sets of analyses for each type of measurement. To specify the measurement you want to analyze, update the `feat_choice` to 'ads' or 'news' in the definition of `load_choice` accordingly.

2.3.1 **[2]** Describe your experimental design in words. Specify what are the treatments and measurements in your experiment?

2.3.2 **[7]** To compute the p-value, the permutation test specifies that we first compute a statistic over the measurements, say $S(\vec{y})$, where $\vec{y}$ represents the measurements. Then we consider all possible permutations of the measurements - $\pi(\vec{y})$ - and compute the same statistic over each of the permutations $S(\pi(\vec{y}))$. The p-value is the proportion of the number of times $S(\vec{y})$ is as extreme as $S(\pi(\vec{y}))$. [8] Note that AdFisher uses accuracy as its statistic - the proportion of the number of correctly labeled units. If there are b testing blocks and $2n$ units in each block (n experimental and n control units). Each browser has an actual label (experimental or control) as well as a predicted label (experimental or control). If $x\%$ of the $2nb$ browsers are correctly labeled, the computed statistic is x. What is the minimum value of the p-value as a function of b and n?

In AdFisher, the default split is done so that first 80% of the blocks are used as training blocks and the last 20% are used as testing blocks. Also, note that AdFisher uses 10-fold cross-validation over the training blocks, so it requires at least 10 training blocks (i.e. $\lceil 10/0.8 \rceil = 13$ total blocks) in the experiment. What is the number of agents ($2n$) and blocks (b) you are using in your experiment? For this selection, using the formula derived above, compute the minimum p-value you can obtain? For a significance level of 0.05 (i.e. you reject the null hypothesis only when the p-value is less than 0.05), is your choice of n and b good enough?

2.3.3 **[7]** Modify the hw3.py script in the appropriate sections to develop the script to run your experiment.

2.3.4 **[4]** Run your experiment and report the test-accuracy and the p-value. Is the p-value significant?

[8] Refer to Section 6 of the Methodology paper in the recommended readings

3 Online Tracking [45=18+4+8+15 points]

For this question, we will learn how to enhance user privacy on the web and make observations on online trackers with some freely available tools. Thanks to Jonathan Mayer for the questions.

3.1 Privacy tools [18=3+3+3+3+3+3]

Technology vendors have taken a variety of approaches to counteracting consumer tracking. This question asks you to explain several of the mostly widely deployed solutions. For each subpart, please provide a concise explanation of the given approach: How does it work? Which tracking methods is it and is it not effective against? Are there any side effect advantages or drawbacks? You may need to do some online reading to answer, especially subparts 3.1.5 and 3.1.6.

3.1.1 [**3**] Adblock Plus [9].

3.1.2 [**3**] NoScript.

3.1.3 [**3**] Tor Browser Bundle [10].

3.1.4 [**3**] Private browsing mode in the web browsers Chrome, Firefox, and Safari. Please mention how Safari's approach differs.

3.1.5 [**3**] Third-party cookie blocking in the four major web browsers. Please mention how the defaults vary in Chrome, Safari, and Firefox.

3.1.6 [**3**] Do Not Track.

3.2 Cookies on Safari [4=2+2]

Safari users are often surprised to find third-party tracking cookies have been placed in their browsers.

3.2.1 [**2**] Explain one way this might happen.

3.2.2 [**2**] Suggest a modification to Safari cookie blocking that would solve the problem. Please do not duplicate the approach of another browser vendor.

3.3 Tracking Measurements on the web[8=6+2]

For this question, you will make tracking measurement on `cnn.com`.

3.3.1 [**2**] How many and what sorts of third parties do you expect will track you on this site?

[9]`https://adblockplus.org/`
[10]`https://www.torproject.org/`

3.3.2 **[6]** Conduct a quick measurement of trackers on the website using a browser-plugin. I find these plugins helpful: Ghostery [11] (for Firefox, Chrome, Safari, Opera), Disconnect [12] (for Chrome), Lightbeam [13] (for Firefox). Also take note of what cookies are being set on your browser by going into your browser preferences. You may want to remove all existing cookies prior to taking this measurement. How many and what sorts of third-party websites are tracking you on the site? Compare what you find from the cookies placed on your browser with what the plugins detected.

3.4 Measurements on Gizmodo Media Group [15=2+3+7+3]

Gizmodo Media Group operates a network of widely read blogs such as Jezebel, Gizmodo, Kotaku, and io9. Each blog is operated from a different domain, e.g. `jezebel.com`, `gizmodo.com`, `kotaku.com`, and `io9.com`. Nevertheless, Gizmodo Media Group is able to collect analytics and offer single sign-on commenting for Safari users across all of its web properties.

3.4.1 **[2]** Before you do any measurement, discuss how you think Gizmodo Media Group might have designed its cross-site analytics. (You do not have to guess correctly—any workable design is a satisfactory answer.)

3.4.2 **[3]** Sketch a methodology for determining how Gizmodo Media Group's analytics work.

3.4.3 **[7]** Follow through on the methodology you described above, and briefly explain the design of Gizmodo Media Group's analytics. Be sure to address the different cases of non-Safari and Safari browsers.

3.4.4 **[3]** Make an argument for and an argument against the proposition that Gizmodo Media Group is circumventing a consumer privacy protection. (You do not need to take a position.)

[11] `https://www.ghostery.com/`

[12] `https://disconnect.me/`

[13] `https://www.mozilla.org/en-US/lightbeam/`

Submission

You have to submit the following files:

- Put all written parts into one pdf file (no word documents) with the name <your_andrew_id>_HW3.pdf.
- Rename the python script file for question 2.3 as <your_andrew_id>_hw3.py.
- The site file you used for question 2.3.
- Rename the logs generated from 2.2 and 2.3 as <your_andrew_id>_log_2.2.txt and <your_andrew_id>_log_2.3.txt.

Put all the files into a zip folder with the name <your_andrew_id>_HW3.zip and submit it through Blackboard. DO NOT submit the entire AdFisher folder.