Obidroid Project Report

Date: Dec 15, 2013

Project Repo: https://github.com/seekshreyas/obidroid

Team

Team Members	Contact
Luis Aguilar	luis@berkeley.edu
Morgan Wallace	morgan.m.wallace@gmail.com
Shreyas	shreyas@ischool.berkeley.edu
Kristine A Yoshihara (from Info 219 Computer Security contributed Research)	100kristine@berkeley.edu

Project Goals

Original intent

As outlined in the Project Proposal, our original intention of project goals was:

> *"Our main aim is develop a **filter** system for flagging unfair apps, via customer reviews and app descriptions."*
>
> *"We do not aim to **predict** if an app is **unfair** or **not** using the reviews/description. Instead we aim to help **scale down** the problem of policing every app periodically on the app store by building a **good indicator/flagging** system."*

So although we built **classifier** for predicting an app is **fair** / **unfair**, we were more interested in coming up with **most informative features** for such a classifier system that could help *scale down* the problem of policing the app store.

Please refer our initial premises of Acceptable Ground Rules, Assessment Scenarios and Deliverables as outlined in the Project Proposal

How far we got

We were able to develop/build:

Module/Utility	Type	Description
`crawler.py`	Script[1]	Given a saved `HTML` page [2] of the Google Play Store app listing eg: free-biz.html, it compiles a list of all the apps listed on that page and exports to a `txt` file in the `inputs/` directory.
`scraper.py`	Script[1]	Scrapes all the desired features of each app url from the crawler's exported txt file and compiles them into their respective `json` files in the `exports/` directory.
Labeling Machinery Scripts	Module	To enable a human to label large quantity of apps manually quickly as **fair**/ **unfair**
Sentiment Analysis Scripts [3]	Module	To aggregate a sentiment score for the overall review for a particular app
Application Data CRUD Operations/Classifier/Analysis Scripts	Module	Scripts for Data Storage and Retrieval, Supervised and Unsupervised learning of the dataset

Labeling Machinery Scripts

One of the challenges we faced during the project was finding good **ground truths** for labeling the apps. Initially were were hopeful that we would be able to obtain such a list from the `FTC`, but when it did not come to fruition, we decided to manually label the apps based on some Evaluation Criteria.

So we developed scripts for our `Labeling Machinery`. These were a collection of scripts that would enable concatenation of given `json` files, export them to a `csv` file that could then be loaded up into softwares like `Excel`, to enable a human to review the desired features in each app and label them.

`300+` apps with `1800+` reviews were reviewed by a human i.e. Morgan Wallace on his Evaluation Criteria to weed out the unfair apps based on user app reviews. This gave a list of 42 apps.

To get the latest features[4] then we tried scraping those apps again, but found that `18` of those had been taken down[5]. So, in a way it did seem to justify our intuition of `unfair` apps labeled just using user reviews.

Scripts	Script Description
`scripts/concat.py`	concatenate the given `json` files
`scripts/json_extract_to_csv.py`	export the overall `json` files to `csv` to enable loading in `Excel` for **labeling**
`scripts/extract_unfair.py`	extract apps labeled `unfair` and append them to `docs/malapps.txt`

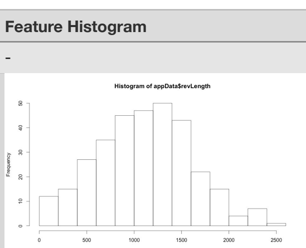
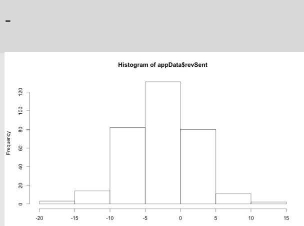
Sentiment Analysis Scripts

Using our Sentence Sentiment Classifier[3], we classified each sentence into `-1, 0, 1` score and then aggregated that score for every sentence in each review. We *chose* **not** to normalize these aggregate scores for the length of the review as we felt that the length of the review was a good bias for our feature.

Scripts	Script Description
`parser.py`, `extractor.py`	for extracting NLP features from each sentence
`mySentClassifer.pickle`	sentence sentiment classifier

Application Data CRUD Operations/Classifier/Analysis Scripts

Storing, retrieving and examining the scraped attributes of the apps we extracted app features and fed them to a `Naive Bayes Classifier` over **4 folds** as **training** and **test data**.

Features Extracted	Feature Description	Feature Intuition	Feature Histogram
`price`	price of an app	Free apps might be more malware ridden	-
`revLength`	total sentences in all user reviews	longer the review, more coherent the user feeling about the app	
`avgRating`	average rating of the app	higher rated apps might be more reliable	-
`hasPrivacy`	whether the app has a privacy policy or not	FTC inspired	-
`revSent`	aggregate review sentiment	NLP inspired	
`hasDeveloperEmail`	app has an associated developer email	FTC inspired	-
`hasDeveloperWebsite`	app has an associated developer website	FTC inspired	-
`countMultipleApps`	app has multiple apps associated with it	self	-
`installs`	average install of each app	self	-
`exclamationCount`	count of exclamation for extreme reviews	NLP inspired	-
`countCapital`	count capitalized words in a review	NLP inspired	-
`adjectiveCount`	count the number of adjectives in	NLP inspired	-
`positiveWordCount`	count the number of positive words from a curated list	NLP inspired	-
`negativeWordCount`	count the number of negative words from a curated list	NLP inspired	-
unigrams like `has(word)`	presence of curated `malindicator` words	NLP inspired	-
`bigrams`	top 20 bigrams via likelihood ration measure	NLP inspired	-
`trigrams`	top trigrams based on raw frequency	NLP inspired	-

Classifier Notes:

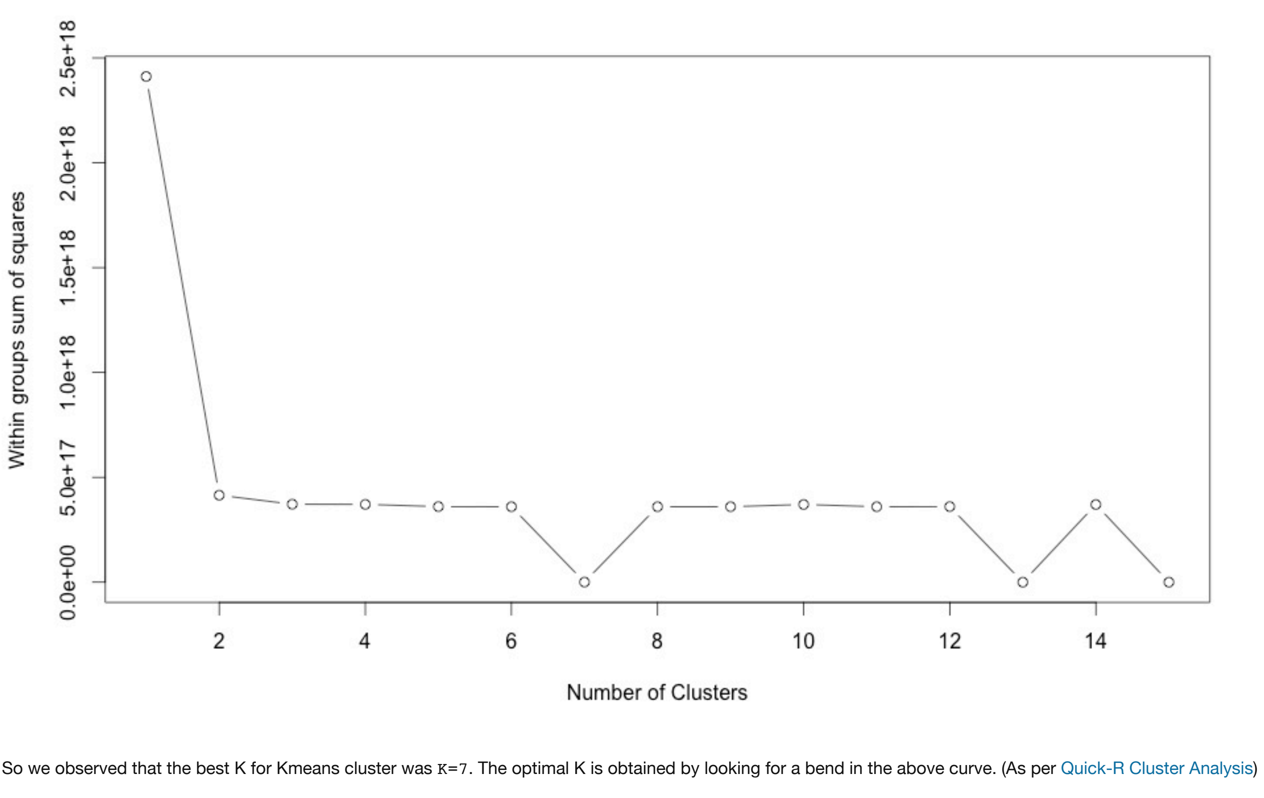
- Although scraping gave us separate counts for each rating (1star, count), (2star, count) …, which we intuitively felt would give use more granular feature for predicting malware, the classifier gave best output for an overall average rating.
- adding `unigrams`, `bigrams` and `trigrams` made our classifier a lot worse (from `~90% -> ~10%`). We eventually turned them off. Please review our rationalizations in Results.

Unsupervised Learning

We did some unsupervised learning with the extracted features by exporting the apps and their features to a `CSV` file and using `R` scripts to do the analysis.

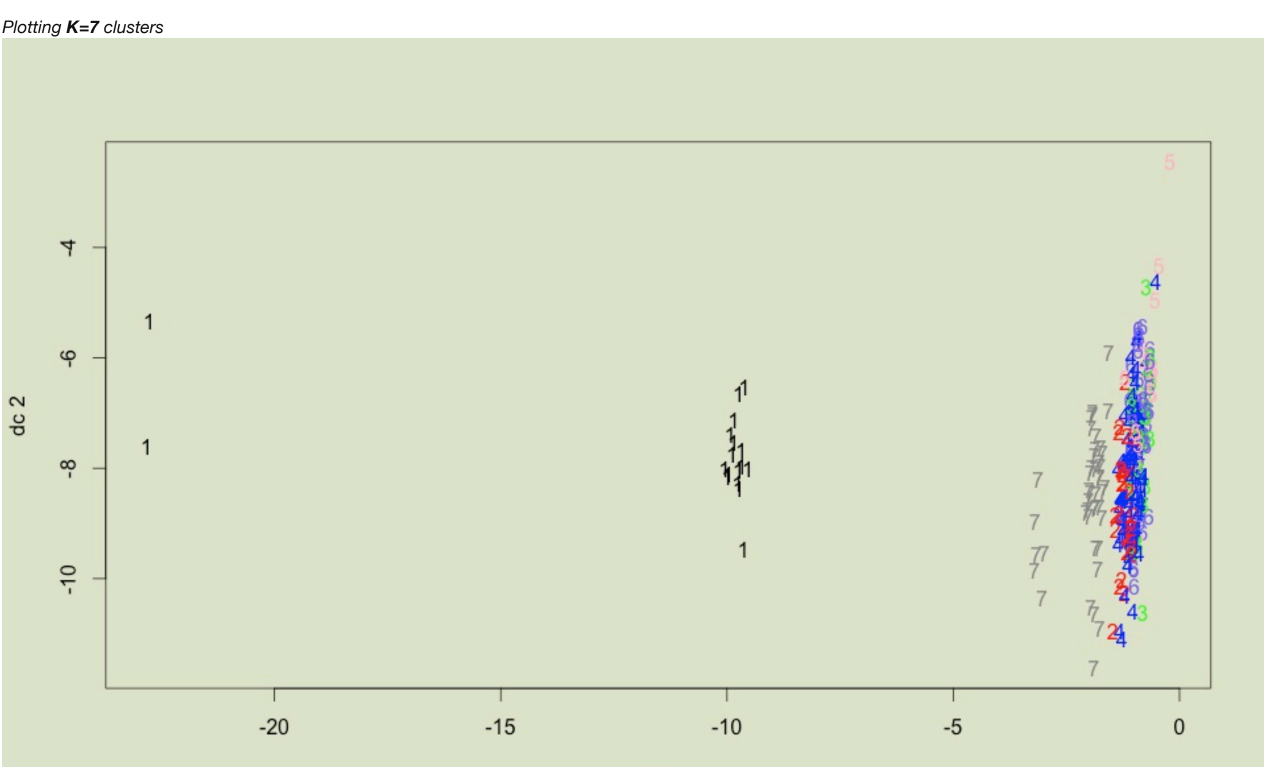
KMeans Clustering

Ideal Number of Clusters (Optimal K)



So we observed that the best K for Kmeans cluster was `K=7`. The optimal K is obtained by looking for a bend in the above curve. (As per Quick-R Cluster Analysis)

*Plotting **K=7** clusters*



It suggests that apps in **cluster1** are a lot further apart than all other apps. **Cluster7** is also markedly different from others. But then the rest of the clusters are quite close together.

Scripts	Script Description
`classifier.py`	extract features from apps and classify them
`db/models/app.py`	SQLAlchemy class that provides communication to app table
`db/models/review.py`	SQLAlchemy class that provides communication to the review table
`db/putAppsReviews.py`	loads db with json files of application attributes and reviews
`db/getAppsReviews.py`	gets from db application attributes and reviews by category or appid
`dataExport.py`	export the app features to CSV which can be loaded in `R` for some analysis
`ranalytics.r` [6]	unsupervised learning on the dataset
`ranalytics_all.r` [6]	full `R` Worksheet

Results / Evaluation

Taking our best results for the classifier, which was cross-validated over 4 folds, :

- Average Prediction Accuracy : `86.71875%`
- Predictions in each fold: `[0.8125, 0.90625, 0.84375, 0.90625]`
- Overall Most Informative Features:
  - `installs`
    - for `installs = 3000.0` the `unfair : fair` ratio was ~ `9 : 1`
    - for `installs = 30000.0` the `unfair : fair` ratio was ~ `6 : 1`
    - for `installs = 3000000.0` the `fair : unfair` ratio was ~ `2 : 1`
  - `revSent`
    - for `revSent = -17` the `unfair : fair` ratio was ~ `8 : 1`
    - for `revSent = -10` the `unfair : fair` ratio was ~ `2 : 1`
  - `countCapital`:
    - for `countCapital = 9` the `unfair : fair` ratio was ~ `3 : 1`
  - `revLength`:
    - for `revlength = 800+` the `unfair: fair` ratio was ~ `2 : 1`
  - `avgRating`:
    - ambiguous

We got an accuracy of `~86%` for `fair/unfair` app prediction. But we would like to mention that we are a little ambivalent about the accuracy as we had few training examples for malapps.

But we would say that we were able to come up with a list of **most informative features** which could be indicators of malapps and hence essentially scale down the problem of evaluating each and every feature of apps and their reviews.

Also, we would like to point out that our sentiment classifier performed fairly well in predicting malapps, showing up in **most informative features** in every fold, we observed that usual NLP features like `unigrams`, `top bigrams`, `top trigrams` performed poorly and brought down the classifier accuracy. And our rationalization for this we believe is related to the ambiguity between **bad reviews** and **unfair reviews**, which is harder to put down in exact words/phrases but is easier to gauge from overall sentiments of the app. Crude NLP features like `countCapital`, which was a count of capital words in a review, were better indicators than bigrams/trigrams.

What future work would be if this were continued

We must obtain more labeled data from additional sources in order to improve our feature selection and classifier.

Adding additional features that take more processing time like: verifying that privacy policy links and developer urls are active and not blacklisted, lightweight parsing of privacy policies for certain keywords, email domain lookups.

Setting up a server to automate this classification over a broader set of the Google Play apps. The automated process will allow newer applications and application version updates to be reviewed and making the results available. Furthermore, the automation needs to take into account that comments are continually added to the application profiles over time. The sentiment of each app will need to be reanalyzed on an iterative basis.

Description of Data

Data was scrapped from the web page specific to each app on the Google Play store. We collected data from 60 applications for each of the following categories:

1. Business (free)
2. Comics (free)
3. Communications (free)
4. Lifestyle (free)
5. Social (free)

We were looking for mostly free apps because a higher proportion of them have higher installs and free apps might be more enticing and malware ridden.

Attributes	Description
`Name`	The official name of the application (e.g. Gmail)
`Company`	The name of the company (e.g. Google)
`AppCategory`	Category of app (e.g. Communication)
`AppId`	The ID used by Google Play to uniquely identify each app (e.g. com.google.android.gm)
`AppVer`	Number representing how many releases of this app there have been (e.g. 2.1)
`Price`	How much the app costs (US Dollars)
`Rating`	Number of 1, 2, 3, 4, and 5 star ratings given by users
`Total Reviewers`	Total number of users that reviewed the app for all versions
`CountOfScreenShots`	Number of screenshots shown by developer
`Installs`	How many devices have this app installed
`ContentRating`	How long the app has been in the store (e.g. Low Maturity)
`SimilarApps`	A list of other AppIDs for apps considered similar by Google Play
`Description`	Developer written description of the app
`MoreAppsFromDev`	A list of AppIDs for other apps that the developer has also made
`User Reviews`	A list of the 6 reviews shown on the page
`Developer Website URL`	Web site link to the developers own page
`Developer Email`	Email address where users can contact the developer
`Privacy Policy URL`	URL of the privacy policy that explains how the developer uses the users` information

Description of Algorithms

Other Contributions

- **Sentiment Analysis Scripts** [3]: was developed with support from **Sayantan Mukhopadhayay**, **Charles Wang** during the earlier assignment.
- **R scripts** [6]: Were borrowed from R blogs and sites like Quick R
- **K-fold validation script**: was referred from: Stack Overflow
- **All other scripts**: were coded in by the team.

Contributions of each team member

- Luis Aguilar
  - Created Postgres database for storage of app information
  - Created SQLAlchemy scripts which moves app JSON data to/from the database
  - Web Service creation for access to app information
  - Development of some of the features in the classifier script
  - Searches for malware apps and FTC/resource correspondence
- Morgan Wallace
  - Crawler to get app IDs and URLs from an HTML page listing hundreds of apps.
  - Conversion script from JSON to CSV for app features for the purpose of labelling
  - Manual labeling ('fair' or 'unfair') of over 1800 app reviews.
- Shreyas
  - Project Architecture
  - Scraper to get all features
  - Feature Extraction Classifier Scripts
  - R Scripts for unsupervised learning
- Kristine A Yoshihara
  - Research, Experiment Design and Statistical Analysis

Code

The entire code is hosted at Github in the Obidroid Repository

Conclusion

Several key features stood above the rest.

- Installs and capital letter usage in reviews were good predictors of malware.
- Surprisingly, average rating from the user was ambiguous. According to our training set, having a high average rating did not preclude an app from being labelled as unfair.
- Lastly, sentiment was a fair indicator, and performed better than other word features like bigrams and trigrams, which could possibly be attributed to words or phrases being poor indicators of malapps as most of the times even users don't know if it is a malware, and it is harder to disambiguate between bad apps and unfair apps.

The work done proves the viability of natural language processing's applicability to detecting malware apps when utilized in conjunction with classification using other application features. We believe it provides a valuable tool for those like the FTC to more efficiently scrutinize the plethora of applications that could be harmful to consumers.

Appendix

- Appendix A: Results
- Appendix B: Code Documentation
- Appendix C: Evaluation Criteria

Footnotes

---

1. Refer to Code Documentation for understanding how to run these utilities with appropriate command line flags. ↩
2. We chose to provide the saved HTML page instead of the live url because the app list is **lazy-loaded** via AJAX and on fetching the live url doesn't give all the apps of a page. ↩
3. We chose to use the sentiment classifier that was created during the sentence classification assignment as the classifier was also trained on user reviews of products. ↩
4. We had to rescrape those app attributes because we had improved our scraper for more in the meantime ↩
5. See the app urls starting with # in `docs/malapps.txt` ↩
6. The R scripts that were used were generally referred from online tutorials and blogs. Majorly, QuickR and InstantR ↩

Appendix A: Results

Best Result

Below are the results from the *best* result output

Accuracy : [0.8125, 0.90625, 0.84375, 0.90625]

```
Train Data
===============================================================================
291
Test Data
===============================================================================
32
Most Informative Features
                installs = 3000.0          unfair : fair   =      9.9 : 1.0
                 revSent = -17             unfair : fair   =      8.0 : 1.0
                installs = 30000.0         unfair : fair   =      4.7 : 1.0
                 revSent = -10             unfair : fair   =      3.4 : 1.0
            countCapital = 9               unfair : fair   =      3.4 : 1.0
               avgRating = 4.256           unfair : fair   =      2.6 : 1.0
               avgRating = 3.987           unfair : fair   =      2.6 : 1.0
               avgRating = 4.522           unfair : fair   =      2.6 : 1.0
               avgRating = 4.424           unfair : fair   =      2.6 : 1.0
               revlength = 1210            unfair : fair   =      2.6 : 1.0
None
Train Data
===============================================================================
291
Test Data
===============================================================================
32
Most Informative Features
                installs = 3000.0          unfair : fair   =      9.5 : 1.0
                 revSent = -17             unfair : fair   =      7.8 : 1.0
                installs = 30000.0         unfair : fair   =      6.2 : 1.0
                 revSent = -10             unfair : fair   =      3.3 : 1.0
            countCapital = 9               unfair : fair   =      2.9 : 1.0
          adjectiveCount = 2               unfair : fair   =      2.7 : 1.0
               avgRating = 4.256           unfair : fair   =      2.6 : 1.0
               avgRating = 3.987           unfair : fair   =      2.6 : 1.0
               avgRating = 4.522           unfair : fair   =      2.6 : 1.0
               avgRating = 4.424           unfair : fair   =      2.6 : 1.0
None
Train Data
===============================================================================
291
Test Data
===============================================================================
32
Most Informative Features
                installs = 3000.0          unfair : fair   =      9.9 : 1.0
                 revSent = -17             unfair : fair   =      7.9 : 1.0
                installs = 30000.0         unfair : fair   =      5.2 : 1.0
            countCapital = 9               unfair : fair   =      2.8 : 1.0
                installs = 3000000.0         fair : unfair =      2.8 : 1.0
                 revSent = -10             unfair : fair   =      2.6 : 1.0
               avgRating = 3.987           unfair : fair   =      2.6 : 1.0
               avgRating = 4.522           unfair : fair   =      2.6 : 1.0
               avgRating = 4.424           unfair : fair   =      2.6 : 1.0
               revlength = 1210            unfair : fair   =      2.6 : 1.0
None
Train Data
===============================================================================
291
Test Data
===============================================================================
32
Most Informative Features
                installs = 3000.0          unfair : fair   =      9.9 : 1.0
                 revSent = -17             unfair : fair   =      7.9 : 1.0
                installs = 30000.0         unfair : fair   =      5.2 : 1.0
            countCapital = 9               unfair : fair   =      3.0 : 1.0
                 revSent = -10             unfair : fair   =      2.6 : 1.0
               avgRating = 4.256           unfair : fair   =      2.6 : 1.0
               avgRating = 3.987           unfair : fair   =      2.6 : 1.0
               avgRating = 4.522           unfair : fair   =      2.6 : 1.0
               avgRating = 4.424           unfair : fair   =      2.6 : 1.0
               revlength = 855             unfair : fair   =      2.6 : 1.0
None
```

Result with NLP Features like unigrams, bigrams and trigrams

Accuracy : [0.125, 0.03125, 0.125, 0.125]

```
Train Data
===============================================================================
291
Test Data
===============================================================================
32
Most Informative Features
                   game. = True            unfair : fair   =     24.6 : 1.0
              consistent = True            unfair : fair   =     22.8 : 1.0
                 charged = True            unfair : fair   =     22.8 : 1.0
                    nuts = True            unfair : fair   =     22.8 : 1.0
                    sold = True            unfair : fair   =     22.8 : 1.0
                 deletes = True            unfair : fair   =     13.6 : 1.0
                    age. = True            unfair : fair   =     13.6 : 1.0
                      tv = True            unfair : fair   =     13.6 : 1.0
  (u'unknown', u'error') = True            unfair : fair   =     13.6 : 1.0
                 factory = True            unfair : fair   =     13.6 : 1.0
None
Train Data
===============================================================================
291
Test Data
===============================================================================
32
Most Informative Features
              consistent = True            unfair : fair   =     19.6 : 1.0
                 charged = True            unfair : fair   =     19.6 : 1.0
(u'great', u'app', u'!') = True            unfair : fair   =     19.6 : 1.0
                    nuts = True            unfair : fair   =     19.6 : 1.0
                    why. = True            unfair : fair   =     19.6 : 1.0
                 viruses = True            unfair : fair   =     19.6 : 1.0
                    lock = True            unfair : fair   =     19.6 : 1.0
                    sold = True            unfair : fair   =     19.6 : 1.0
                   virus = True            unfair : fair   =     16.4 : 1.0
       contains("virus") = True            unfair : fair   =     16.4 : 1.0
None
Train Data
===============================================================================
291
Test Data
===============================================================================
32
Most Informative Features
                   game. = True            unfair : fair   =     22.2 : 1.0
              consistent = True            unfair : fair   =     20.5 : 1.0
                 charged = True            unfair : fair   =     20.5 : 1.0
                    nuts = True            unfair : fair   =     20.5 : 1.0
(u'great', u'app', u'!') = True            unfair : fair   =     20.5 : 1.0
                    why. = True            unfair : fair   =     20.5 : 1.0
                 viruses = True            unfair : fair   =     20.5 : 1.0
                   admin = True            unfair : fair   =     20.5 : 1.0
                  secure = True            unfair : fair   =     20.5 : 1.0
                    lock = True            unfair : fair   =     20.5 : 1.0
None
Train Data
===============================================================================
291
Test Data
===============================================================================
32
Most Informative Features
              consistent = True            unfair : fair   =     22.8 : 1.0
                 charged = True            unfair : fair   =     22.8 : 1.0
                    why. = True            unfair : fair   =     22.8 : 1.0
                 viruses = True            unfair : fair   =     22.8 : 1.0
               contacts. = True            unfair : fair   =     22.8 : 1.0
                    lock = True            unfair : fair   =     22.8 : 1.0
                   virus = True            unfair : fair   =     19.1 : 1.0
       contains("virus") = True            unfair : fair   =     19.1 : 1.0
                    safe = True            unfair : fair   =     19.1 : 1.0
                   game. = True            unfair : fair   =     19.1 : 1.0
None
```

Appendix B: Code Documentation

More documentation to come. For now

Classifier

Run classifier using :

```
$ python classifier.py --dir="exports/"
```

Scraper

Run scraper using:

```
$ python scraper.py --dir="inputs/" --export="exports/"
```

Crawler

Run crawler using:

```
$ python crawler.py --input="<path.to.htmlfileofappstore>"
```

Evaluation Criteria

1. Misuse of personal data
2. Privacy policy
3. Developer email
4. Low ratings

Look for:

- “spam”
- questions to developers about permissions and security, like “why do you need my…”
- “spam”
- “virus”
- mentions of “privacy policy”
- “personal info/information”
- “spying”
- “access (your|my)…”
- language around “warning” other users
- Mentions of “facebook”, “contacts” and other permissions-related items.
- “fake”
- “lies | liar(s)”

Fair apps:

- mentions:
    - love
    - great
    - Excellent
    - useful
    - perfect
    - Amazing
- General good sentiment

Bad but fair apps:

Many apps are bad but don’t actually act maliciously

- bugs & “bug fix”
- “can’t”
- “no longer work”
- “error”
- Dissapointed
- upgrade
- "“iphone port”"
- “slow”
- “UI” | “Layout” | “GUI”
- “crash”
- “does not work”
- “broken”| “flaw(s)”
- “tweaks”
- negative “quality”
- “useless”
- expensive
- “bring back”
- Mention of type of phone - usually relates to bug fix or incompatibility
- “please help”
- “It would be better if”
    - if it were malware, they wouldn’t try to help the developer improve it by giving helpful feedback

Other recommendations for classification

just classify using review titles since they are brief and to the point.