Bot Detection System Architecture

The bot detection system is intended to give a web service provider the ability to automatically detect bots with a reasonable amount of certainty even if the bot is not identified explicitly.

At the centre of the system is the Apache mod_ml module. This module serves two functions: formatting and forwarding data on user behaviour for preprocessing and, secondly, as an intermediary in the classification process. Figure 1 illustrates these two functions.

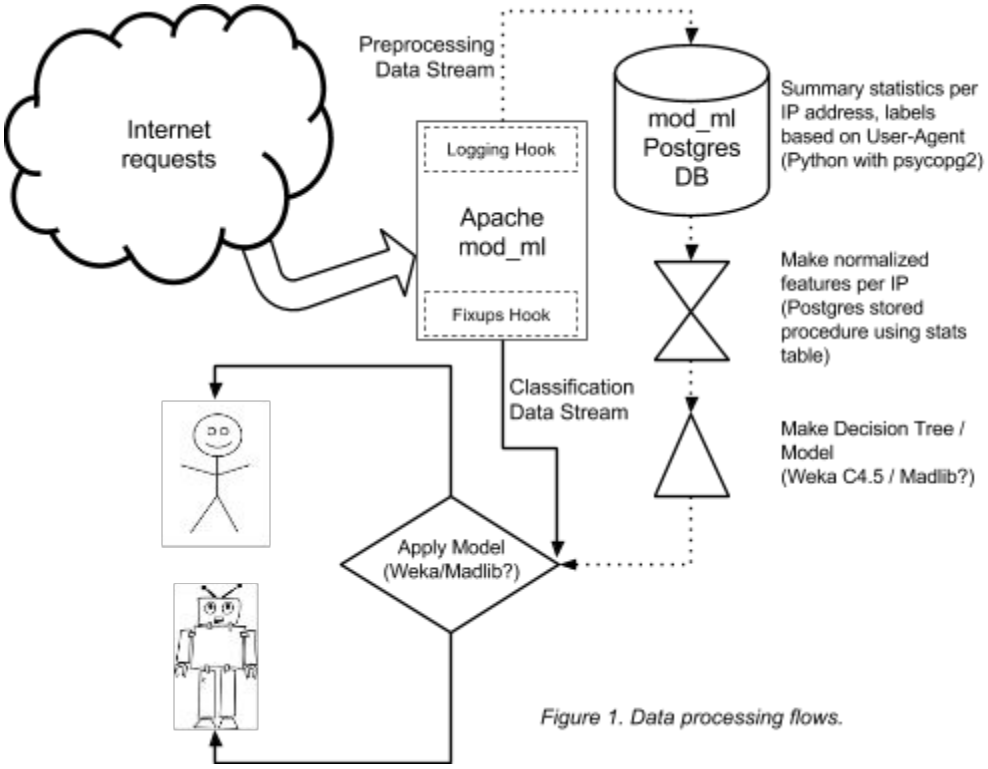

Figure 1. Data processing flows.

Preprocessing

The preprocessing function will be implemented via the mod_ml MLPreProcess directive which will forward JSON encoded request data in the form:

```
{"useragent":"xxx", "epoch":"1437481437361", "hour":"12", "REMOTE_ADDR":"45.74.2.131",
"REQUEST_URI":"/ottawa.craigslist.ca/ths/5108776913.html", "HTTP_HOST":"206.12.16.219:8898",
"status_line":"200 OK"}
```

Where the keys are

- **useragent** - the contents of the User-Agent header; this, classified as bot or non-bot by the Google http://www.useragentstring.com API, is used to train the model; the data is cached in the mod_ml Postgresql database located on cloudsmall8 *botlabels* table

- **epoch** - the epoch time in milliseconds from the apache request_rec data structure; differences in time are used to generate features for model building
- **hour** - the hour of the request; once again differences in time are used to generate features for model building
- **REMOTE_ADDR** - the IP address of the requester; this, along with the HTTP_HOST, is used to group the stats and for classification
- **REQUEST_URI** - the URI after the domain:port; this is used to determine the type of content being downloaded; the ratio of html vs non-html content is used as a feature
- **HTTP_HOST** - which web server served the request; used with the ip as a key for grouping stats
- **status_line** - the HTTP return code for the request; the ratio of 200/300 vs 400/500 is used as a feature for model building

Data for each REMOTE_ADDR/REQUEST_URI pair is saved in two tables in the mod_ml DB: *botlog* and *botiplabels*:

The botlog table

Column Name	Column Type
blid	integer primary key
hour	integer
remote_addr	character varying(128)
status_line	character varying(64)
useragent	character varying(255)
epoch	bigint
request_uri	character varying(255)
http_host	character varying(128)

The botiplabels table

Column Name	Column Type
http_host	character varying(128)
remote_addr	character varying(128)
label	character varying(32)
isbot	integer
ip	character varying(128)

The “*isbot*” and “*label*” fields are mapped to the *botlog.useragent* via the *botlabels* table

Column Name	Column Type
useragent	character varying(255)
label	character varying(32)
isbot	integer

The current list of labels from the Google www.useragentstrng.com service is along with their the arbitrary classification used for identifying bots for supervised learning (classification from the author):

User-Agent Label	Bot?
Mobile Browser	no
Browser	no
Offline Browser	no
unknown	no
Console	no
Cloud Platform	no
Feed Reader	yes
Librarie	yes
LinkChecker	yes
Crawler	yes
Validator	yes
Other	yes

The *botlog* table keeps one or more records of request activity for a given *HTTP_HOST*, *REMOTE_ADDR* pair. In theory only one previous record is required as subsequent records are combined together into the botstats table:

Column Name	Data type	Notes
ip	character varying(64)	HTTP_HOST/REMOTE_ADDRESS
n	integer	count of items saved in diffs column
sum	bigint	sum of items saved in diffs
mean	double precision	mean of diffs items
var	double precision	variance of diffs items
skew	double precision	asymmetricality of diffs distribution
kurtosis	double precision	flatness of diffs distribution (more -ve = flatter)
diffs	integer[]	list of paired epoch time differences
pages	bigint	number of html page requests
reqs	bigint	number of requests of any type
hourdiffs	integer[]	pairwise differences based on hour of the request
hn	integer	count of hourdiffs
hsum	bigint	sum of hourdiffs
hmean	double precision	average hour difference
hvar	double precision	variance of hourdiffs
hskew	double precision	asymmetricality of hourdiffs distribution
hkurtosis	double precision	flatness of hourdiffs distribution

The diffs and hourdiffs arrays are allowed to grow to 1000 items in length but the pages and reqs counts are allowed to grow to arbitrary sizes. With the scheme only the most recently seen html file request needs to be saved in the database, greatly reducing the required space.

This summary data is generated by a network based service that receives the JSON data from mod_ml, looks up the HTTP_HOST/REMOTE_ADDR in the *botlog* table (the *botlog.ip* field). If that *botlog.ip* field already exists in the database, it processes the new data and replaces the row. If that ip field does not exist, a new row is created.

To make data that is more amenable to model building a pl/pgsql stored procedure “botnormalize” was developed to make a table of normalized features. Normalization in this case reduces the values to a range of between 0 and 1 by dividing each with the min - max difference for that field for all entries in the *botstats* table. The resulting table has the following structure:

Column Name	Data Type	Notes
ip	character varying(128)	key to botstats
isbot	integer	designation from botiplabels
mean	double precision	normalized epoch mean
var	double precision	normalized epoch variance
skew	double precision	normalized epoch skew
kurtosis	double precision	normalized epoch kurtosis
hmean	double precision	normalized hour mean
hvar	double precision	normalized hour variance
hskew	double precision	normalized hour skew
hkurtosis	double precision	normalized hour kurtosis
poverr	double precision	normalized proportion of pages to requests
uacount	double precision	number of user agents associated with this ip

This table will be generated periodically via cron on cloudsmall8 to refresh the model.

Model building

Classification

Future work

At the moment, how the normalized data is to be transformed into a model including what features are the most important for distinguishing bots and non-bots is not known.

A three month set of Apache log data has been acquired and a replay script has been developed. A subset representing traffic over three two-day periods for two websites was used to initially test the system and is being used for development work. For testing other short time periods will be used. Unlabelled data will be classified and the classification compared with the www.useragentstring.com designation of the User-Agent header for the requests. Ideally most

of the known bots will be identified. As some bots may not identify themselves with User-Agent strings it is expected that there will be at least some “false positives.” As distinguishing false positives from obscured bots is impossible the overall success of the system cannot be precisely determined.

Some other data from the request_rec struct could be used - such as the mime type of the requested item. At the moment stats need to be generated based on the status_line as well.