Social Sense: Quantifying Website Intent

Joel Shackelford, Alexander Spizler, Yang Xu

{jshack02 , aspizler , yxqc4} @ stanford.edu

Data Conditioning

Common Crawl Corpus

The Common Crawl corpus contains petabytes of data collected over the last 7 years. It contains raw web page data, extracted metadata and text extractions. The data is made available in Web ARChive (WARC) format. An example is shown below:

```
WARC/1.0
WARC-Type: response
WARC-Date: 2014-08-02T09:52:13Z
WARC-Record-ID:
Content-Length: 43428
Content-Type: application/http; msgtype=response
WARC-Warcinfo-ID:
WARC-Concurrent-To:
WARC-IP-Address: 212.58.244.61
WARC-Target-URI: http://news.bbc.co.uk/2/hi/africa/3414345.stm
WARC-Payload-Digest: sha1:M63W6MNGFDWXDSLTHF7GWUPCJUH4JK3J
WARC-Block-Digest: sha1:YHKQUSBOS4CLYFEKQDVGJ457OAPD6IJO
WARC-Truncated: length

HTTP/1.1 200 OK
Server: Apache
Vary: X-CDN
Cache-Control: max-age=0
Content-Type: text/html
Date: Sat, 02 Aug 2014 09:52:13 GMT
Expires: Sat, 02 Aug 2014 09:52:13 GMT
Connection: close
Set-Cookie: BBC-UID=...; expires=Sun, 02-Aug-15 09:52:13 GMT;

<!doctype html public "-//W3C//DTD HTML 4.0 Transitional//EN"
<html>
<head>
<title>
        BBC NEWS | Africa | Namibia braces for Nujoma exit
</title>
...
```

Amazon Web Services

The data is read in from a public Simple Storage Service (S3) bucket and processed using MapReduce jobs on Elastic Compute Cloud (EC2) instances.

Intermediary results and final feature vectors are emitted and stored in S3 buckets. All intermediaries are deleted when a batch process finishes as there is a cost associated with persistent storage.

amazon web services | S3 | EC2 | Elastic MapReduce

Methods

Our objective, determining which sites are likely to link to Facebook, is a safe (though trivial) substitute for another problem; predicting whether a website contains a link to a malicious website. We use the following methods to classify websites into one of two categories depending on whether the website contains a link to Facebook.

Logistic regression

We first use Newton's Method to explore a small, manually extracted, data set (88 samples/5 features). Figures 1 and 2 show some features that appear to separate our classes. With more data, we decided to use stochastic gradient ascent, as it is more efficient with large data sets and more feature vectors (53,779 samples/267 features).

SVM

We use the lib-linear package to train different models for our classification problem. Also we use the data-scaling tool provided by lib-linear to scale our feature data. Since some of the features have significantly higher value than others, the data-scaling allows us to rescale all the features to have relatively close values, thus our prediction accuracy is improved based upon the same large feature vectors.

Another SVM tool we use is the lib-svm, which uses non-linear mapping of the features but runs much slower than lib-linear. For both SVM tools, we use the default parameters.

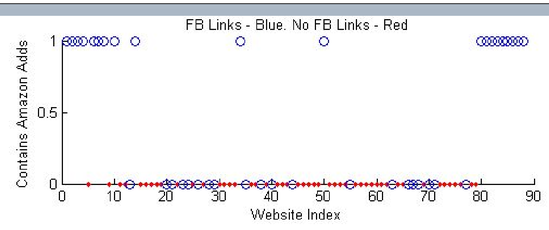

Figure 2: This plot appears to show that sites with links to Facebook are more likely to have larger page sizes and longer retrieval times

Features	Description	Data type
UUID	Unique ID for each web page	String (e.g. 663ff1d7-2ea2-4255-9f95-102536d625d0)
Number of links on a page	Sum of all links given in the HTTP response	Integer
Page size	As expressed in the HTTP metadata	Integer
Contains amazon adds	True if site has Amazon Ads	Boolean
Extends a Google API	True if site has Google user info/sign in, maps, etc.	Boolean
Contains an expiration	As expressed in the HTTP response	Boolean
Domain extension	e.g. .com, .gov, .edu and various county codes	261 element Boolean vector
Server type (future goal)	80 most common servers (e.g. Apache, Microsoft, etc.)	80 element Boolean vector
Retrieval time	Time, in ms, from HTTP request to response	Integer
Raw text vectorization (future goal)	Utilizing Google's word2vec pretrained vocabulary	300 element Boolean vector

Table 1: List of current and desired features we extract from each website

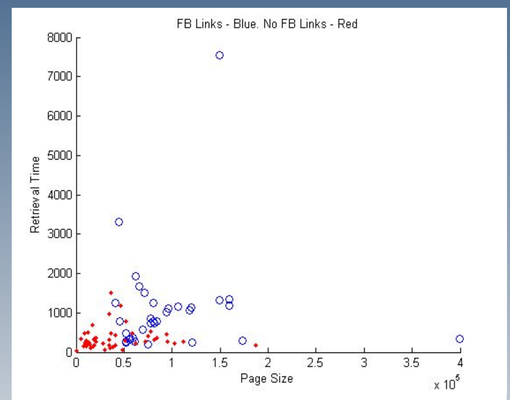

Figure 1: This plot appears to show that sites with links to Facebook are more likely to have larger page sizes and longer retrieval times

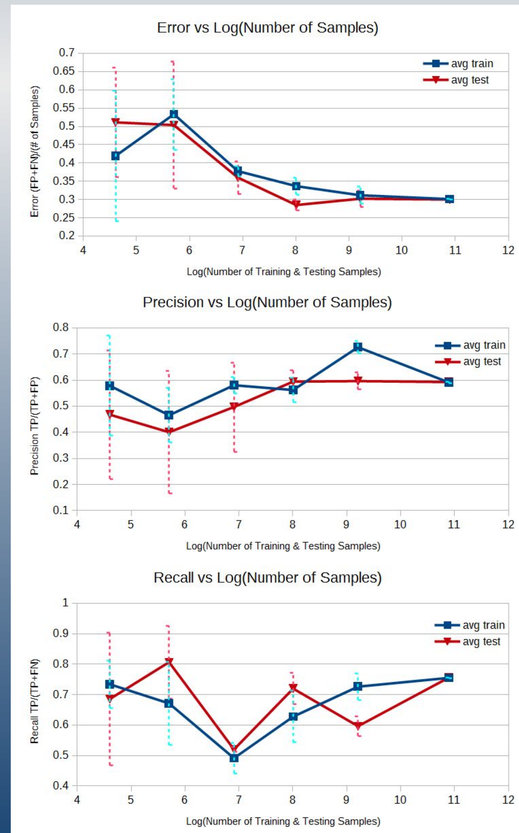

Figure 3: Logistic regression with growing datasets shows that error (top), precision (middle), and recall (bottom) all converge on testing and training data

Results

Large similar errors in both testing and training data (~30%) lead us to believe that we have a larger than desired bias, but do not have a problem with overfitting. A method to overcome bias is to add more features, so we would like to use features derived from the text content of each website. We plan on utilizing Google's word2vec to represent overall sentiment of a given web site.

Feature	Theta
x0 (offset)	-140.23
number of links	27847.90
Page size	-26.77
Contains Amazon ads	10.28
Extends Google API	42.71
Contains expiration	-47.77
Domain	
.com	-61.35
.org	-32.91
.net	-6.34
.gov	-9.13
.edu	-12.09
.mil	-0.06
.int	-0.50
.co	-1.52
.ac	-1.76
Retrieval time	-398.46

Table 2: Values of theta after logistic regression with scaled features.

Conclusions

For lib-linear and lib-svm, training by using the scaled data lead to better accuracy. E.g, for lib-linear, the training error on raw feature data gave us accuray as 64.44%, while scaled-data can achieve 71.67%.

The lib-linear method showed issue of overfitting, with testing accuracy noticeably lower than training error. However, logistic regression and lib-svm have not shown overfitting, with their training and testing accuracies close to each other.

Training Method	Training Accuracy	Testing Accuracy
LR	69.89%	70.01%
LR - Scaled	70.08%	69.94%
Liblinear - Scaled	71.67%	60.98%
Lib SVM - Scaled	65.58%	65.41%

Table 3: Training and testing accuracies on different learning algorithms