Scalable Spam Classifier for Web Tables

Santiago Villasenor, Tom Nguyen, Anusha Kola, Sean Soderman, Michael Gubanov
Department of Computer Science
University of Texas at San Antonio

***Abstract*—Internet mail spam is a problem for most organizations and individuals. Spam can be classified into two categories: fraud and commercial. The fraud category includes phishing, scams, malware, counterfeit products and any other criminal activities. The commercial category includes promotional messages and newsletters that we do not want to receive, being sent illegally from legitimate organizations. Fraud can be seen as being a high threat with high volume while commercial spam is the opposite. Similar to mail, there are spam Web tables that do not have any useful content. Here we describe our machine-learning classifier for efficient and effective Web tables spam filtering that was tested on a large-scale Web tables corpus of $\approx$ 36 million tables.**

***Keywords*-Web-search; Large-scale Data Management; Cloud Computing; Data Fusion and Cleaning; Summarization; Human-Computer Interaction.**

I. Problem Statement

Although spam has seen a sharp decrease in recent years [Kaspersky Lab,], it is still a major problem that stands in the way of technical and nontechnical individuals alike. For everyday users, the systems in place for Gmail and MSN for spam filtering is sufficient most of the time.

However, the cleanliness of data extracted from the Web itself varies. Not only do many "empty" webpages filled with advertisements exist, many instances of structured Web tables with useless or sparse data also exist, presenting a problem for those who wish to analyze or browse the data inside of them. An example of this would be if a data scientist extracted a large-scale corpus of Web tables from the Internet. For example, if they wanted to use this data to build a word co-occurence matrix, they would need to identify the spam in this structure and repeat the analysis of the dataset with this algorithm after ignoring the spam words they identified. Eventually, they may come up with a set of hard-coded rules to remove spam in their dataset. Since every data scientist would have to do this for each of their specific applications, the amount of time wasted in total would be huge. In addition to this, such manual cleaning of data is error-prone and could still leave spam data behind.

With the above problem in mind, it is necessary to filter out spam from Web table data. Automatic filtering of spam data will not only make life easier for the previously mentioned data scientists, but also for those experimenting with search over these datasets. Such a tool would need to be constructed in a different manner than one made for tasks such as e-mail spam classification, as spam data looks drastically different. An example Web tables spam follows:

Listing 1. A sample of Web tables spam

```
"null null null null null null null null null null
     null null null null null null null null null
    null null null null null null null null 0 ",1
""images" null null null null null "64" "29" "27"
    "87" "48" null null "54" "58" "40" "20" "29"
    null null null null null null null null null
    null null 0 ",1
"quantity display_error_msg_div(na
    eventquantity2_msg_10435669);
    display_error_msg_div(na
    eventquantity2_msg_10435699); ...
    display_error_msg_div(na
    eventquantity2_msg_10425011);
    display_error_msg_div(na
    eventquantity2_msg_10435619);",1
" &nbsp;&nbsp; gmb (new) ",1
"arent&nbsp;they&nbsp;cool?&nbsp; ",1
" the record order is not maintained between the
    list and form for multiple edit ",1
" 18.30 kevin tunstall 18130 0",1
" nobility found without conceit?&nbsp;|where is
    there ",1
" y/c ",1
```

In Listing I, the data was pulled from several rows of our Web tables corpus and is sparse with information. This is much different from the spam common to email and other media. Thus, a new approach to spam classification is required in this context.

II. Solution

We used a custom scalable Naive Bayes Multinomial Text classifier for Web tables spam classification. The Naive Bayes classifier is a probabilistic classifier based on the Bayes theorem with strong and naive independence assumptions. This classifier is very useful in applications such as email spam detection, personal email sorting, etc. Another benefit is that the classifier is very efficient for unstructured and semi-structured data like in Web tables crawled from the Web. Fortunately, we have a significant amount of training data from our corpus. This will also aid in the effort to make the classifier scalable as it will require less resources with large data sets compared to other algorithms. We have gathered results from the ZeroR and Input Mapped Classifier algorithms, but determined that

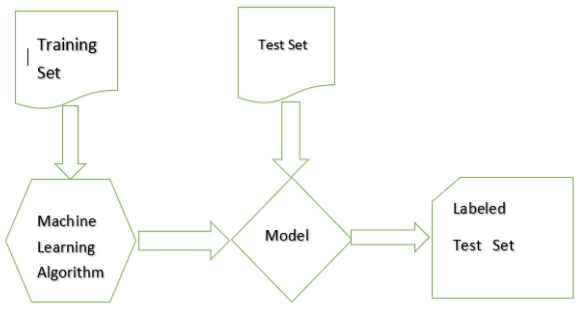

Figure 1. Training and classification of Web tables spam

our custom scalable Naive Bayes Multinomial performed better with our current training set.

The training data supplied to the classifier algorithm is labeled as spam/non-spam. Figure 2 illustrates training and classification pipeline for Web tables spam.
The algorithm for generating the training data is as below.

1) Execute the listed queries for the positively labeled training set.
2) Label positively each selected row.
3) Execute the inverse of these queries for the negatively labeled training set.
4) Lebel negatively each row.

Queries: In a large-scale structured Web tables corpus with ≈ 36 million tables, we have encountered a significant amount of spam. After observing certain common patterns, we came up with the queries that can be used to generate the spam training data:

- Column1/Attribute1 is not null while the rest of the columns are null
- The total tuple length is more than 400 characters
- The total tuple length is less than 40 characters
- The entry contained "&nbsp" which means space in HTML
- Other HTML formatting rules

To apply the above rules to our dataset, we used queries similar to the one below:

Listing 2. Sample queries generating spam training data

```
select * from spam_corpus where
len(column2) > 120 and
column3 like '%&nbsp%' and
column4 is NULL and
column5 is NULL and
column6 is NULL and
column8 is NULL
```

```
select * from spam_corpus where column2
    is NULL and
column3 is NULL and
column4 is NULL and
column5 is NULL and
column6 is NULL and
column8 is NULL
```

```
select * from spam_corpus where
len(column2 .. column11) +
len(column13) > 400
```

```
select * from spam_corpus where
len(column1 .. column10) < 40)
```

```
select * from spam_corpus where column1
    like '&nbsp' and column2 is null
```

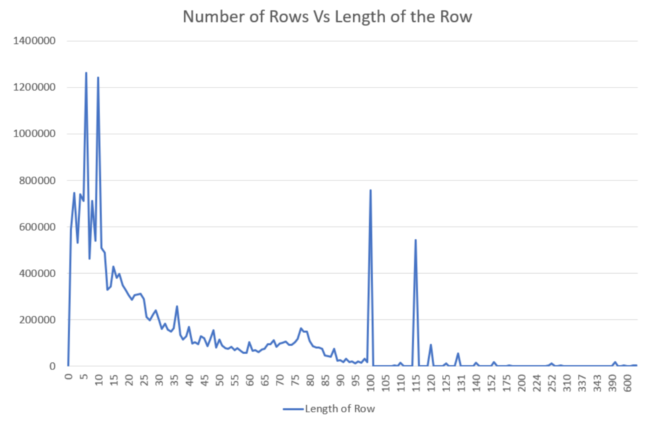

Figure 2. Number of Spam Tuples vs. the Tuple Length

III. Evaluation

The graph above illustrates the statistics of the spam rows that were classified by our trained spam classifier. It has labeled 21 million rows as spam from the corpus. The X-axis in the graph provides the length of the spam row and the Y-axis gives the number of rows observed for the corresponding length.

We observed 92.2% precision and 91.3% recall using 10-Fold Cross-Validation. The training set was 54 thousand rows ≈ 50% positive and 50$ negatively labeled instances. The detailed evaluation of our custom Naive Bayes Multinomial Text Classifier for Spam is below in tables I , II, and III.

Performance: We believe that we can further improve our precision by improving our training sets, re-evaluating our search criteria, modifying our SQL queries that generate training data, and evaluating results from multiple machine learning algorithms. As of now we have increased our training data to ≈ 55,000 entries.

Table I
OVERALL PERFORMANCE

Metric	Value
Correctly Classified Instances	91.3257%
Incorrectly Classified Instances	8.6743%
Kappa Statistic	73.02%
Mean absolute error	11.87%
Root mean squared error	26.9%
Relative absolute error	39.7075%
Root relative squared error	69.5714
Total Number of Instances	54967

Table II
DETAILED ACCURACY BY CLASS

Metric	Class 0	Class 1	Weighted Average
TP Rate	86%	92.5%	91.3%
FP Rate	7.5%	14%	12.8%
Precision	72%	96.7%	92.2%
Recall	86%	92.5%	91.3%
F-Measure	78.4%	94.6%	91.6%
MCC	73.5%	73.5%	73.5%
ROC Area	94%	94%	94%
PRC Area	87.5%	98%	96%

IV. RELATED WORK

[Cafarella et al., 2008] developed a system to work with a large-scale dataset of Web tables from Google web crawl. The dataset was originally 14.1 billion tables, and mostly contained non-relational tables such as tables for page layout, extremely small tables, or tables used for laying out calendars. They used parsers to remove the 89% of tables with these properties, afterwards training a classifier on human-labelled tables containing a large number of blank cells, simple lists in two dimensions, and tables containing pairs associating attributes to values.

Here we omit the details of crawling, extracting, and cleaning up the large-scale Web tables dataset that we have. We forgo the parsing process completely, opting for queries we constructed that are highly indicative of spam data in the corpus. We use a machine-learning classifier to rule out data with undesirable properties. However, our training data is composed of the result sets from the aforementioned queries to train our spam classifier, which contrasts with the human-labeled sets of data they use for training. In addition to this, our training set is based on the distribution & sparseness of content rather than structural properties such as lists or attribute-value pairs. Finally, their classifier is ruling out entire *tables* at a time, while we focus on individual *rows*. In this way, *we can retain all useful information from tables that are partially spam*.

[Eberius et al., 2015] construct a large-scale dataset similar to [Cafarella et al., 2008], the key difference being that they focus on classifying the *layout* of web tables in order to improve the amount of web tables recovered from the Web. This is important because many papers that focus on the task of Web table detection assume that web tables are structured in the same manner. That is, tables are laid out as 2-dimensional structures with attribute-column headers and relations under them, similar to a table in a relational database.

Table III
CONFUSION MATRIX

n = 54957	Predicted as 0	Predicted as 1
Actual 0	8648	1409
Actual 1	3359	41551

While the layout of web tables in our own corpus may vary, we focus primarily on the *content* of tables rather than their *structure*. Our goal of identifying spam tables is also different from the objective of finding tables stored in nonstandard ways.

[Ntoulas et al., 2006] focus on the task of identifying spam web pages. They mention two techniques, *link stuffing* and *keyword stuffing* as ways some website owners might try to increase the rank of their websites. Link stuffing is the practice of creating web pages with many links to a page they wish to be ranked more highly. Keyword stuffing is where website creators place extraneous words, sometimes in large amounts, in certain parts of a web page. They focus on content-based heuristics such as the number of words in the title, as well as average word-length on a page.

Although link stuffing has no analog in the context of web tables, keyword stuffing is something that we could have considered during our work on this project. However, it cannot be said that tables containing a large number of even the same word are necessarily spam. For example, a historical table documenting several women with the title "lady" may be identified as spam in the context of web pages. These titles are likely necessary as they could disambiguate these people from others with the same name. Since the context of relational web tables is different from that of Web search, we had to identify several different spam criteria from web table data such as the data in Listing 1.

REFERENCES

[Cafarella et al., 2008] Cafarella, M. J., Halevy, A. Y., Zhang, Y., Wang, D. Z., and Wu, E. (2008). Uncovering the relational web. In *WebDB*.

[Eberius et al., 2015] Eberius, J., Braunschweig, K., Hentsch, M., Thiele, M., Ahmadov, A., and Lehner, W. (2015). Building the dresden web table corpus: A classification approach. In *Big Data Computing (BDC), 2015 IEEE/ACM 2nd International Symposium on*, pages 41–50. IEEE.

[Ntoulas et al., 2006] Ntoulas, A., Najork, M., Manasse, M., and Fetterly, D. (2006). Detecting spam web pages through content analysis. In *Proceedings of the 15th international conference on World Wide Web*, pages 83–92. ACM.

[Kaspersky Lab,] Kaspersky Lab. Global spam volume as percentage of total e-mail traffic from january 2014 to september 2017, by month. Statista. Accessed 15 Nov 2017.