Parsing Domain WhoIs Information with Different Patterns Using One Generic Parser

Jiale Tan, Xing Li, and Yiqing Xing

(SUIDs: jiale, xingli, and xingyq)

Abstract

We design a generic parser to read the output file from more than 600 whoIs Servers all over world. There is no uniform and standard format for these files, and formats also change over time. This leads to huge maintenance cost for firms who regularly harvest these files and read in data using regular expression. In this project, we design a generic parser that is robust to changes in format in different sources and over time. With a training sample of about 1000 files, our parser results in smaller testing errors as we are making use of more sample files.

Introduction

There are more than 600 whoIs servers[1] worldwide who are providing the whoIs information of the domains[2]. Lots of firms in different industries make use of whoIs information in their business. For example, Apple is interested in finding out who has registered ipad.com, Nike would like to know who owns the online marketplace which sells counterfeits of their products, and Reuters can take advantage of whoIs information to support their news, etc. For these firms, one of the routine jobs is to regularly submit queries to whoIs servers that will send back a string, and then read the string and parse out useful information.

Informative strings from whoIs servers have different formats both within and across servers, and the format also changes over time. The variation in formats among servers causes a huge maintenance cost for user firms to use standard regular expression to read in information, because they need to detect and test different patterns for more than 600 whoIs servers and in addition make modifications when format changes, which is also unpredictable. More specifically, an average whoIs server has 4-5 patterns, and the pattern changes once every 12 weeks, which will induce to about 15,000 regular expression patterns in a year.

[1] For example, whois.godaddy.com.

[2] whoIs information include registrar, registrant/tech/admin's name/organization/email/phone/fax/address, name server, status

In this project, we try to design a generic parser to read whoIs strings. Although our parser is not as precise as the regular expression, it is robust enough to cater for different formats among different servers over time.

To express the idea in an abstract level, let $(x^{(t)}, y^{(t)})$ be the training sample, where $X^{(t)}$ is a string of J words $X^{(t)} = (x_1^{(t)}, x_2^{(t)}, \dots, x_J^{(t)})$, and $y^{(t)} \in X^{(t)}$ is the information we want to parsed out. The parser will assign a probability $p_j(X^{(t)})$ to each of J words in string and pick out the word with the highest probability to be the information. To decrease dimensionality, we restrict our probability function to be a function of M words before, i.e.,
$p_j(X^{(t)}) = p(x_{j-1}^{(t)}, x_{j-2}^{(t)}, \dots, x_{j-M}^{(t)})$.

whoIs strings

The objective we are dealing with is the whoIs strings from more than 600 servers all over world. They have different patterns within one server as well as between servers, and they changes constantly over time. For example, the following are two strings with totally different formats.

Record 1:
…
Domain Name: DANDYLION-RECYCLE.COM
Registrar: PURENIC JAPAN INC.
Whois Server: whois.purenic.com
Name Server: DNS001JPS.SUNFIRSTASIA.NET
Name Server: DNS2.SUNFIRSTASIA.NET
Registrant Name: Hidetoshi Fujita
Registrant Organization: DANDY LION
…
Record 2:
…
Domain isabellabrendolin.com
DNS1: ns1.serverlet.com
DNS2: ns2.serverlet.com
Registrant
Isabella Brendolin
Email:domain@serverlet.com
Via Corsica 3
35016 Piazzola Sul Brenta
Italy
Tel: +39.3495424998Email: elalex928@yahoo.com.hk
…

In our algorithm, we assign probability to each word predicting whether this is the field we are interested. For example, in record 1, we assign probability to each of the words “Domain”,

"Name", "DANDYLION-RECYCLE.COM", "Registrar", …., and pick up the one with the highest probability. If the model predicts correct, the probability assigned to "DANDYLION-RECYCLE.COM" should be highest.

For simplicity, we restrict our probability to be dependent only on M words before. Suppose $M = 2$, in our example of Record 1, p("DANDYLION-RECYCLE.COM") = f("Domain", "Name"), and p("Registrar") = f("Name", "DANDYLION-RECYCLE.COM")

Learning models

Converging to email spam problem

Our problem is isomorphic to the email spam problem after pre-processing to the format of email spam. Specifically, for a file tokenized in a string of words, we stack that into multiple observations where each observation consists of M words as feature and whether the word immediately after is the content to be detected as categories. See figure 1 for an example.

File (t)						
\	\	A	B	C	D	…
$x_1^{(t1)}$	$x_2^{(t1)}$	$x_3^{(t1)}$	$z^{(t1)}$			
	$x_1^{(t2)}$	$x_2^{(t2)}$	$x_3^{(t2)}$	$z^{(t2)}$		
		…	…	…	…	
$y^{(tj)} = 1\{z^{(tj)}$ is Domain Name$\}$						

$$x^{(t)} = \begin{bmatrix} x^{(t1)} \\ x^{(t2)} \\ x^{(t3)} \\ x^{(t4)} \\ \dots \end{bmatrix} = \begin{vmatrix} \text{X} & \text{X} & \text{Domain} \\ \text{X} & \text{Domain} & \text{Name} \\ \text{Domain} & \text{Name} & : \\ \text{Name} & : & \text{DYND} \\ \dots & \dots & \dots \end{vmatrix} \qquad y^{(t)} = \begin{bmatrix} y^{(t1)} \\ y^{(t2)} \\ y^{(t3)} \\ y^{(t4)} \\ \dots \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ \dots \end{bmatrix}$$

Problem Candidate Models

a) Naïve Bayes Multi-Nominal Model:

A multi-nominal Naive Bayes model can be imposed to pin down the probability function described above, say,

$$f\left(x^{(tj)}|y^{(tj)};\phi\right) = \prod_{k=1}^{M} f\left(x_k^{(tj)}|y^{(tj)}\right) = \prod_{k=1}^{M} \phi_{x_k^{(tj)}|y=y^{(tj)}}$$

b) Order-specific Multi-Nominal Model

One of the questions of the model above is the order invariance of the prediction probability, which will make the prediction of "Domain Name" being the same as "Name Domain". One modification is to allow probability to be order-specific, say

$$f\left(x^{(tj)}|y^{(tj)};\phi\right)=\prod_{k=1}^{M}f_k\left(x_k^{(tj)}|y^{(tj)}\right)=\prod_{k=1}^{M}\phi_{x_k^{(tj)}|y=y^{(tj)}}$$

c) SVM

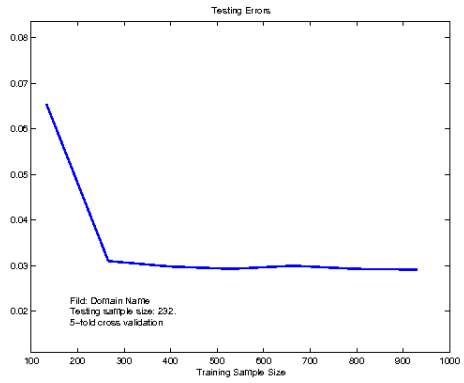
At the end of day, we will go back to SVM to pursue the best prediction.

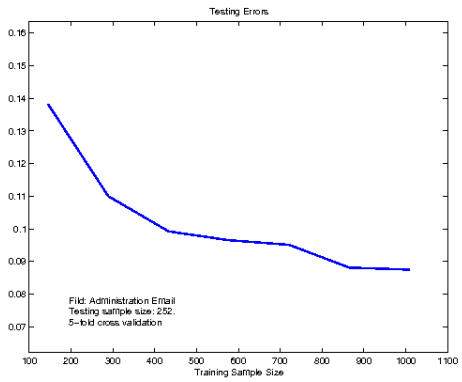
Other Implementation Details

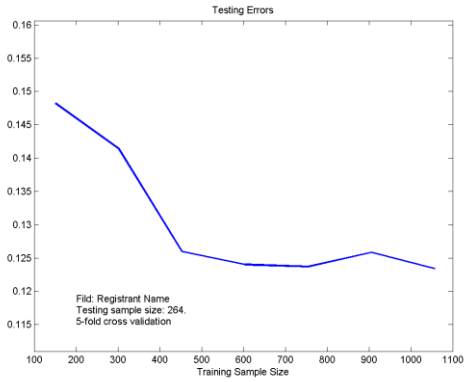
- We drop all the starting irrelevant words and start our string from "domain".
- We pick the dictionary of the words by taking the union of frequent words (80% of all words) appeared within M words before the field of interest.

Testing Results

We plot the learning curve using 5-fold cross validation within which we fix the testing sample and allow the training sample to grow. We report the learning curve of three fields of interest: domain name, administration email and registrant name. The learning curve is decreasing pretty fast, and as sample size goes, we believe our parser will work pretty well after enough training.







Next Steps

There is a lot of room to reduce the error and improve performance:

1) Preprocessing:

a) We can pick features through filtering out irrelevant words

2) Modeling:

a) We can pick the maximum likely set of proceeding keywords for corresponding fields

b) We can use the *n* tokens instead of single tokens as keys in our dictionary so that we can take advantage of the order of the tokens in the feature windows

c) We can optimize the number of words we want to look in a row before or even use flexible token size based the proceeding keyword

3) ML Algorithm:

a) We can try SVM which might reduce the error rate.