Internet Traffic Classification Using a Markov Model and Kullback-Leibler Divergence

Jinsoo Hwang*
Dept. of Statistics Inha University, Incheon, Korea - jshwang@inha.ac.kr

Jeankyung Kim
Dept. of Statistics Inha University, Incheon, Korea - jkkim@inha.ac.kr

Kichang Kim
School of Communication & Engineering, Inha University, Incheon, Korea - kchang@inha.ac.kr

Abstract

As internet traffic rapidly increases, fast and accurate network classification is becoming essential for high quality of service control and early detection of network traffic abnormalities. Machine learning techniques based on statistical features of packet flows have recently become popular for network classification partly because of the limitations of traditional port and payload-based methods. In this paper, we propose a Markov model-based network classification with a Kullback-Leibler divergence criterion. Our study is mainly focused on hard-to-classify patterns of network applications, which current techniques have difficulty dealing with. The results of simulations show the superiority of our suggested method.

Keywords: traffic classification; Markov model; machine learning; traffic pattern; Kullback-Leibler.

1. Introduction

Traditional methods of network classification are either port-based or payload-based. However, both types of approaches are currently facing numerous challenges from the advanced techniques used to circumvent the firewalls of organizations and the increasing number of packet encryption techniques. Machine learning-based network classification techniques that use statistical information from network traffic data are currently emerging.

While several Markov models and hidden Markov models are used to extract information from packet sequence and directions, we utilize the Markov model proposed by Munz et al. We also implement the "bag of flow" concept proposed by Zhang et al. to handle correlated traffic efficiently.

In this paper, we use a Markov model with states defined by the direction and size of the first few packets. In the training stage, we build and train a Markov model for each application. In the testing stage, we first form a group of connections based on port number only, and then build a Markov model with a given group size n, after which we use Kullback-Leibler divergence to measure the divergence between the Markov models of the training set and those of the test set. Finally, we classify a test group to an application whose Markov model has the smallest divergence with that of the test group. We verify the performance of our proposed method by means of theoretical and real data simulations.

2. Background

In network traffic classification port-based approaches classify packets based on the ports used by the packets. However, nowadays, many applications, especially P2P applications, use unpredictable or dynamic ports, which limits the efficacy of this approach. Payload-based, or Deep Packet Inspection (DPI), approaches look deep inside the packet to capture its application-specific pattern. However, this technique also suffers from challenges such as frequently changing packet formats, encryption of the packet payload, and load increments. Several means of improving supervised learning techniques have been attempted. Nguyen and Armitage proposed taking packet samples from various locations during packet transmission to build multiple sub-flows. Most classification techniques either capture the entire flow or the first few packets of the flow as samples, but they claim that capturing the entire flow is time-consuming and detecting the beginning of packet transmission is not always possible. Instead, they capture packets in an intermittent way during packet transmission and use these multiple sub-flows to train their modeling system. Some other efforts have taken into consideration the time series characteristics of the transmitted packets in addition to their statistical properties. Dainotti et al. and Mu and Wu constructed Hidden Markov Models (HMMs) to represent the traffic classes while Munz et al. built Markov Models with eight states and four stages, which is much simpler than building HMMs but still effectively expresses the time series and statistical characteristics of the transmitted packets. [Zhang] proposed constructing a bag of flows and classifying it as a whole instead of classifying individual flows. Classifying a bag of flows as a whole has the advantage that the portion of the mis-classified flows can be ignored if it occupies a smaller percentage of the bag, because the larger portion of the correctly classified flows determines the membership of the entire bag.

Our technique combines the Bag-Of-Flow (BOF) technique with the Markov model approach. We believe the Markov model is simple and powerful enough to grasp the characteristics of the traffic and the BOF concept is essential to improving the classification accuracy. However, applying BOF directly to packet classification does not always produce the best results, especially when the target traffic classes show similar and overlapping characteristics, such as SMTP and IMAP packets. In such a case, direct application of BOF actually results in a more inferior performance than individual assignment (the details are given in later section). We propose the construction of another Markov model for the flows contained in the bag and to measure its similarity with the target Markov models. The flows in this bag are all assigned to the traffic class whose Markov model is the most similar to the test Markov model.

Classification performance can be calculated by using *recall* and *precision*, which are used in [Munz], and F-measure to evaluate per-class performance, as in [Zhang].

- $recall_k = \Pr$ (classified as $k|$ true class k)
- $precision_k = \Pr$ (true class $k|$ predicted as k)
- F-measure $= \frac{2\times precision \times recall}{precision + recall}$= harmonic mean of *recall* and *precision*

3. Classification using Kullback-Leibler Information

In this paper, we focus on two application packets, SMTP (port 25) and IMAP (port 143), whose traffic patterns are difficult to distinguish using existing classification algorithms. Bernaille et al. showed that the first four packets of a TCP connection are sufficient to classify known applications with high accuracy; therefore, we build our Markov model using only the first four packets exchanged. State space is defined by a combination of direction of packets and four payload intervals[1]: [0, 99], [100, 299], [300, MSS-1], and [MSS]. The value of the Maximum Sequence Size (MSS) is often exchanged in a TCP connection. Since the direction is either from client-to-server or server-to-client, each stage can have $4 \times 2 = 8$ different states. Thus, our model becomes a four-stage left-right Markov model with state space $S = \{0, 1, \ldots, 7\}$. States 0-3 represent payload length intervals from client-to-server while states 4-7 represent those of server-to-client. For example, state sequence 0-4-1-4 means the following: client sends a 0-99 byte range packet first (after the handshake), the server responds with a 0-99 byte range packet, the client then sends a little larger 100-299 byte range packet, and finally the server responds with a 0-99 byte range packet.

By investigating the state sequences of SMTP and IMAP in training data, we find that 0-4-1-4 is the dominant pattern in both applications. Other common patterns, such as 1-4-1-4 and 0-4-0-4, also exist in both applications. In this section, we explain our classification model, which has only two common patterns: 0-4-1-4 and 1-4-1-4. We then extend our model in the simulation section to incorporate an extra unique pattern per application.

Suppose that we have two common patterns, 0-4-1-4 and 1-4-1-4, in both applications. Let M_k be the Markov model for application k (App$k$) and two patterns as observations, say O_1(0-4-1-4) and O_2(1-4-1-4), in our Markov models. Assume that p_1 and p_2 are the proportions of O_1 in App1 and App2, respectively. Thus, the initial state probabilities are $\pi_0^k = p_k, \pi_1^k = 1 - p_k, \pi_l^k = 0, \forall l = 2, \ldots, 7$ for each model $k = 1, 2$. That is,

$$\boldsymbol{\pi}^k = (\pi_0^k, \ldots, \pi_7^k)' = (p_k, 1 - p_k, 0, \ldots, 0)'.$$

[1]We follow the same payload intervals as in [Munz]. These intervals have been chosen because they emphasize well the difference in traffic feature vectors among the various applications[Munz].

Since we have four stages, we need three transition probability matrices in addition to the initial probability vector to compute a pattern probability. Let P_{ij}^k denote a one-step transition matrix from stage i to j in M_k, i.e., $P_{ij}^k(s1, s2) = P(M_k(j) = s2|M_k(i) = s1)$. On the basis of the frequency of observations O_1 and O_2 in each model, we can build three transition probability matrices, P_{12}^k, P_{23}^k, and P_{34}^k.
Let O_1 (0-4-1-4) and O_2(1-4-1-4) be the only two observations of Markov models. Thus, for each connection, we can define $l_k(O_1) = P(M_k(1) = 0, M_k(2) = 4, M_k(3) = 1, M_k(4) = 4)$ and $l_k(O_2) = P(M_k(1) = 1, M_k(2) = 4, M_k(3) = 1, M_k(4) = 4)$. We can say $l_k(O_j)$ is the likelihood of $O_j (j = 1, 2)$ under model M_k. Each likelihood is computed as follows:

$$l_k(O_1) = \pi_0^k P_{12}^k(0, 4)P_{23}^k(4, 1)P_{34}^k(1, 4) = p_k$$

$$l_k(O_2) = \pi_1^k P_{12}^k(1, 4)P_{23}^k(4, 1)P_{34}^k(1, 4) = 1 - p_k$$

We propose three group assignment methods: Majority, Kullback-Leibler, and 4096d. Majority assignment is based on the voting of each individual assignment, Kullback-Leibler assignment is our main proposed method, and 4096d assignment can be another reasonable candidate method based on Euclidean distance in 4096 dimensions.

Kullback-Leibler: For each bag of connections, we build a Markov model M^{TE} and measure divergence by computing Kullback-Leibler information $I(M^{TE}; M_1)$ and $I(M^{TE}; M_2)$. For $k = 1, 2$

$$I(M^{TE}; M_k) = \sum_{j=1}^{4096} l^{TE}(O_j) \log \frac{l^{TE}(O_j)}{l_j^{TR}(O_j)}$$

where $l^{TE}(O_j)$ and $l_j^{TR}(O_j)$ are the likelihoods of O_j under the model in test and training data, respectively. Since our Markov model consists of four stages and eight states, we have $8^4 = 4096$ possible observations in test data ($O_j, j = 1, \ldots, 4096$). To avoid division by zero, we use $l_k^{TR}(O_j) = 10^{-5}$ for such j with $l_k^{TR}(O_j) = 0$ and $l^{TE}(O_j) \neq 0$.

If $I(M^{TE}, M_1) < I(M^{TE}, M_2)$, then we assign a group of testing connections to M_1. The evaluation measures *recall* and *precision* can be calculated in a similar fashion to the Majority case.

4096d: We can map each BOF of size n to a point in 4096-dimensional space, because we have $8^4 = 4096$ possible observations. For example, if a size-10 test BOF consists of $4O_1$, $2O_2$, $1O_4$, and $3O_5$, it can be represented as a point in 4096-dimensional space like $(4, 2, 0, 1, 3, 0, \ldots, 0)$. After suitable standardization, we use the Euclidean distance to determine the membership of a test BOF.

4. Simulation Experiments

In the following hypothetical model-based simulations we choose values of p_1, p_2, q_1, and q_2 representing the real traffic sequences of ours and the more challenging scenarios, i.e. $p_2 \approx q_2$. And for simplicity of notations, we use capital letter abbreviations to represent evaluation measures and subscript numbers to denote the application as usual.

- R_k =*recall*$_k$, P_k=*precision*$_k$, F_k=F-measure$_k$, U_k=*undecided*$_k$
- Maj = Majority, K-L = Kullback-Leibler

Case 1: $p_1 = 0.6, \quad q_1 = 0.4, \quad p_2 = 0.5, \quad q_2 = 0.5$

		Evaluation							
Group	n	R_1	P_1	F_1	U_1	R_2	P_2	F_2	U_2
Maj	10	.6331	.6268	.6299	.2007	.3770	.6940	.4885	.2461
	100	.9729	.6789	.7997	.0103	.4602	.9649	.6232	.0796
K-L	10	.6331	.6268	.6299	.0	.6230	.6294	.6262	.0
	100	.8211	.8582	.8393	.0	.8644	.8285	.8461	.0
4096d	10	.6331	.6268	.6299	.0	.6230	.6294	.6262	.0
	100	.8689	.8252	.8465	.0	.8159	.8616	.8381	.0
Individual		.6	.5454	.5714	.0	.5	.5555	.5263	.0

Case 1 represents a situation in which there are only two patterns. As expected, group assignment gives a better performance than individual assignment and both K-L and 4096d are better than Maj. An interesting observation in this case is the fact that the low values for R_2 in individual (50%) and Maj (46%) improve up to 86% in K-L as bag size becomes $n = 100$. Under M_2, which have 50% O_1 and 50% O_2, we wrongly assign up to half of true App2 to App1 in individual and Maj in group assignment. Nevertheless, even in that case, K-L performs well as group size increases. Since similar performance can be achieved under various cases, we omit the rest of our model-based simulation results.

In real data based simulation we retrieved traffic data from the packet traces in [Trace]. Our simulation system used the pcap library functions to extract valid TCP connections from the traces. We defined a valid TCP connection as being a packet exchange between a client and a server that starts with proper three-way TCP handshakes and has at least four packets after them[2]. Among the packet traces, we singled out SMTP and IMAP connections. The trace files were huge and produced over 160,000 connections for SMTP and over 30,000 connections for IMAP.

Table 1 shows the state sequences (patterns) for each application in sorted order with the most frequent at the top. Both applications had the 0-4-1-4 sequence as the most frequent sequence, with the other common sequences being 0-4-0-4, 0-4-0-0, and 1-4-1-4.

Given a real network traffic observation O_j, we calculate the empirical likelihood under each model M_k,

$$\hat{l}_k(O_j) = \hat{\pi}^k(s1)\hat{P}^k_{12}(s1, s2)\hat{P}^k_{23}(s2, s3)\hat{P}^k_{34}(s3, s4)$$

where $\hat{\pi}^k(s1)$ is the proportion of state $s1$ in the first stage and $\hat{P}^k_{ij}(si, sj)$ is an empirical transition matrix constructed from the total N_k connections. Evaluation measure, $recall_1$ can be computed using empirical likelihood $\hat{l}_k$ and $\hat{P}_1(O_j) = n_{1j}/N_1$ instead. The other evaluation measures can be computed in a similar fashion.

Table 1: Composition of state sequence

IMAP		SMTP	
State sequence	Proportion	State sequence	Proportion
0 4 1 4	0.5564	0 4 1 4	0.4611
0 4 1 0	0.2189	0 4 0 4	0.2870
1 4 1 4	0.1134	0 4 0 0	0.0906
0 4 0 0	0.0790	1 4 1 4	0.0687
0 4 1 1	0.0108	1 4 2 4	0.0150
0 4 0 4	0.0108	1 4 0 4	0.0141
....	....	....	....

Table 1 shows why SMTP and IMAP applications are difficult to classify using conventional network classification methods, such as the BOF technique in [Zhang] and a plain Markov model approach in [Munz]. By eyeball computation, $l_{SMTP}(0414) < l_{IMAP}(0414)$ and $l_{SMTP}(1414) < l_{IMAP}(1414)$, so state sequences 0-4-1-4 and 1-4-1-4 of SMTP are mis-classified as IMAP. Therefore, the recall rate of SMTP is at most 1-0.5298.

5. Detecting P2P Packets

Detecting P2P packets such as BitTorrent is a hard problem and has been investigated by numerous researchers. In this section, we describe how our technique can be applied to detect BitTorrent packets and provide some preliminary experimental results. Since our technique needs a set of flows believed to belong to

[2]Currently our system eliminate flows with less than four packets. However, it is not hard to extend our system to handle flows with less than four packets. We can simply add zero-length packets at the end of the flow when it does not contain all the four packets. Applications that produce less than four packets then can be characterized as flows ending with a number of zero-length packets.

the same protocol, in this case BitTorrent, we have collected packets coming out from the same host to build a bag of flow and applied our technique to it. We have identified hosts that exchange packets with more than 10 different peers within relatively short time period, in our case 1000 seconds. There were 1049 such hosts in the pcap files used in the experimentation. Since the pcap files do not contain payload portion, we can not tell exactly which traffic are due to true BitTorrent application[3]. Instead we have assumed port 6881 through 6899 are BitTorrent ports and collected packets with these ports as BitTorrent traffic. Packets with these ports have very high chance of being BitTorrent packets[4], and the purpose of our experimentation is to see how much of them are classified as BitTorrent packets by our KL model.

Table 2: Port distribution of the flows from the 1049 hosts

port	21	22	25	80	110	119	143	443	995	BT	Other
True	273	38	8278	0	136	51	14	755	1	943	26634
Pred	744	145	8036	4626	294	191	189	11115	84	11699	0

The following Table 3 shows the result of our KL model in detecting P2P packets.

Table 3: Prediction result for BitTorrent hosts

	True port		Predicted port	
	num of BT	num of Non-BT	num of BT	num of Non-BT
host 1	25	110	125	10
host 2	30	5	35	0
host 3	75	8	83	0
host 4	48	1	49	0
host 5	14	2	16	0
host 6	27	24	41	0
host 7	62	5	67	0
host 8	8	4	10	2
host 9	13	1	1 4	0
host 10	22	3	25	0
host 11	3	33	10	23
host 12	56	1	50	7
host 13	6	19	10	15
.....	.....	.....	.....	.....
total	943	880	1724	99
recall	1.0000			
precision	.5470			

6. Conclusions

In this paper, we developed a novel classification method based on a Markov model with Kullback-Leibler divergence. Our primary goal was to develop a method that performs well on hard-to-classify cases. Even

[3]Another approach of collecting traffic for BitTorrent detection would be generating BitTorrent packets by ourselves and combine them with existing pcap files. But it is very hard to generate realistic BitTorrent traffic in this way, and thus only very typical pattern of BitTorrent packets are produced. In our laboratory, the BitTorrent traffic generated by our simulated BitTorrent host nodes was detected with our KL-10 model trivially.

[4]Similar approach can be found in [Bern] ...

though most of the previous methods of network classification perform well in most cases by using either correlated information of connections or a combination of a machine learning technique and Markov or hidden Markov models, they fail to produce convincing results when the patterns of connections of applications are similar.
We proposed a novel method that combines a flexible Markov model with Kullback-Leibler information and correlated traffic connection by grouping or bagging with the port number of applications. As our theoretical simulation and real data simulation shows, our method outperformed the other methods in hard-to-classify situations, even though we did not cover all possible cases.
We recognize the slowness of the Kullback-Liebler approach as being one drawback of our technique. However, its high prediction success rate even among the overlapping traffic classes in terms of traffic patterns, as in SMTP and IMAP, is promising. Further, our technique is scalable in that its execution time increases linearly as the number of target classes increases, because once it builds the Markov model for the test data, measurement of its distance from each of the Markov models of the target classes can be done very quickly. The performance of detecting P2P packets show a perfect *recall* rate and reasonable *precision* rate under a rather restricted scenario. So we can extend and test our method under various real situations.

References

Bernaille, L. Teixeira, R., Salamatian, K., "Early application identification," Proc. of ACM International Confernece on Emerging Netwrk Experiments and Technologies (CoNEXT) 2006, Lisboa, Portugal 2006.

Dainotti, A., Pescape, A., and Claffy, K. C., "Issues and Future Directions in Traffic Classification," IEEE Network, January, 2012.

Mu., X., Wu, W., and Enabled C., "A Parallelized Netowrk Traffic Classification Based on Hidden Markov Model," Distributed Computing and Knowledge Discovery, Oct., 2011.

Munz, G., Dai, H., Braum, L., and Carle, G., "TCP Traffic Classification Using Markov Models," TMA'10 Proceedings of the Second international conference, pp. 127-140, 2010.

Nguyen Thuy T. T. and Armitage, G., "A Survey of Techniques for Internet Traffic Classification Using Machine Learning," IEEE Communications Surveys and Tutorials, Vol. 10, No. 4, 2008.

Zhang J., Xiang, Y. Wang Y., Zhou, W., Xiang, Yong, and Guan, Y., "Network Traffic Classification Using Correlation Information," IEEE Trans. on parallel and distributed systems, Vol. 24, No. 1, pp. 104-117, Jan., 2013.

http://www.simpleweb.org/wiki/Traces