Skype Traffic Classification: Naive Bayes or Neural Networks

Mohammad Jalali[*]
Department of Computer Science
University of Toronto
Toronto, Ontario, Canada
mjalali@cs.toronto.edu

Abstract

Skype is one of the largest Voice over Internet Protocol (VoIP) providers with over 500 million users. Skype was created by the developers of KaZaa and unlike many of its competition; it operates on a Peer-to-Peer (P2P) overlay network (It is not using the IETF Session Initiation Protocol (SIP)). Because of its popularity, encrypted traffic and proprietary design, there has been a surge in the research community and the industry to understand its architecture and network traffic. In this paper, I compare two different machine learning classification techniques, Naive Bayes and Neural Networks, to see which one is more successful in identifying Skype traffic from the other types of network traffic.

In order to assess these algorithms, I have designed and developed an application which uses the Operating System's network tools (netstat and tcpdump) to capture and label each of the network flows and then extract statistical features of these flows. These features are used to train and test the classifiers and the labels are used as the ground truth in the evaluation.

1 Introduction

Traffic classification is an essential part of the network traffic management which allows time-sensitive packets meet their performance goals (QoS: Quality of Service) and stops malicious traffic from spreading. One of the most successful approaches to classify the network traffic is using Machine Learning techniques. Other techniques such as port checking and regular expression checking are bound to fail when we have to deal with an application which communicates in an encrypted manner and tries to hide from being detected by generating partially random pattern of traffic (For example, Skype tries to avoid detection, because many service providers may benefit from compromising its traffic).

A reliable Skype network traffic classifier can indicate the significance of this task when considering network operators introducing network policies regarding Skype calls and monitoring Skype's performance. This classifier will allow the network provider to increase the priority of Skype calls during routing and improve the call quality or block Skype due to local legislations (For example, it is illegal to use Skype in United Arab Emirates).

[*]Thanks to James Martens, Inmar Givoni and Richard Zemel for all their help and support during this project.

2 Related Work

Network traffic detection is an emerging field. Different techniques have been developed for identifying and classifying network traffic. These techniques usually fall into one of the following groups:

The first group uses the port numbers on the packets to classify them. For example, the packets that are generated by port 80 are classified to be of type HTTP. As stated by Curtis [8], it has been shown that this technique is not effective at all for Skype and P2P applications that dynamically choose their port number. Kim et al. [11] claim an accuracy of 4-13% for the port-based classifier CoralReef [12].

The second group performs deep packet inspection for patterns which can distinguish a network flow (Similar to matching a regular expression) [4]. These techniques are currently the most common but as described by Moore et al. [9], they cannot be more accurate than 50-70% and when the traffic is encrypted the accuracy plunges even more. One of the famous implementations of this group is BLINC which captures the fundamental patterns of the transport layer and considers three levels of host behavior (social, functional and application levels). The authors of BLINC [10] claim an approximate accuracy of 80-95%. Another problem with this approach is that the pattern signatures have to be developed by the network administrator and can be a tedious and error-prone task.

The third group use Machine Learning (ML) techniques to detect network traffic. There has been a few research papers describing these techniques applied to different types of traffic and these methods are proven to be the most effective for classifying network traffic. Kim et al. [11] compare CoralReef, BLINC and seven ML algorithms and show that these techniques are the most accurate and effective when it comes to classifying encrypted traffic (Accuracy up to 98%). But the majority of these papers focus on the internet traffic as a whole rather than focusing on individual software [11, 13, 14, 15]. A few actually just apply the WEKA library [17] using the default parameters [11, 15, 16]. Trivially, their result can be improved if these parameters are fine-tuned for a specific application. In [2], Bonfiglio et al. use Naive Bayes approach and Pearsons Chi-Square test (packet inspection) together to reach an accuracy of close to 80% in detecting Skype traffic. In addition, they describe various techniques to fine tune the parameters and features to improve the performance. I have not been able to locate any papers applying other ML algorithms to Skype traffic.

3 Classification Model

In this section, I describe the various components for classification. Firstly, I define what a network flow is and how it contains enough information to be uniquely assigned to an application. In the next subsection, I describe the main flow features I adopted in my project and why they are representative of each network flow including Skype. Finally, I briefly describe the classification algorithms and parameters used for their training.

3.1 Network Flows

In order to design a good network traffic classifier, we must first define its basics. As we know, network traffic consists of data packets travelling between a source and destination. Without loss of generality, if we consider a simple client-server model, these packets can be generated by different applications but destined from the same client to the same server. Therefore, we will need a more robust model other than the packet's IP (Internet Protocol) address to identify the application generating them.

In this paper, we define a network flow to be a sequence of packets with the same 5-tupple (source and destination IP addresses, source and destination port numbers and protocol number) which has been active for the last 100 seconds (flows inactive for longer than 100 seconds are assumed to be timed-out and completed).

Note that each flow is unique to each application because the Operating System only assigns each port to a single application at each time and therefore, flows can be used as a unique component to be classified. Moreover, classifying flows would be equivalent to classifying the applications generating the flow.

3.2 Feature Selection

Choosing good flow features is the most challenging and crucial part of designing a classifier. In [9], Moore et al. use Flow duration, TCP Port, Packet inter-arrival time (mean, variance, ...), Payload size (mean, variance, ...), Effective Bandwidth based upon entropy and Fourier Transform of the packet inter-arrival time to classify internet traffic using Bayesian approach. Note that the main focus of this paper is classifying TCP traffic and since most of the Skype flows use UDP protocol and random ports then the TCP port and bandwidth becomes irrelevant. Bonfiglio et al. [2] use message size (audio call packets sizes are different from other traffic) and average inter-arrival time (to detect the fixed rate codec) to classify Skype traffic using Naive Bayes. Therefore, I have chosen to use the following 17 features:

- *Flow duration*
- *Client flow - packets inter-arrival time* (mean, variance, max, min)
- *Server flow - packets inter-arrival time* (mean, variance, max, min)
- *Client flow - packets size* (mean, variance, max, min)
- *Server flow - packets size* (mean, variance, max, min)

The reason for these choices is that Skype traffic has longer flow duration compared to most of the web traffic. Also, because of the use of codecs for audio calls and time sensitivity of packets in Skype, these features can help distinguish Skype traffic.

3.3 Classification Algorithms

As mentioned before, for this experiment, I used Naive Bayes and Neural Networks to classify Skype traffic. Naive Bayes [9, 11, 19] is the simplest probabilistic classifier based on Bayes theorem, which analyzes the relationship between each feature and the application class for each instance to derive a conditional probability for the relationships between the feature values and the class. Neural Networks [11, 20, 21] is a highly interconnected network of units, neurons, whose output is a combination of the multiple weighted inputs from other neurons. I used the most common and simple Neural Network classifier called the Multilayer Perceptron model, which consists of a single input layer of neurons (features), a single output layer of neurons (classes), and one hidden layer between them. The number of hidden units was set to 4 and I set the learning rate (weight change according to network error) to 0.003 and I ran the training for 3000 epochs (an epoch is the number of times training data is shown to the network). Moreover, I utilized my own implementation of these algorithms which I have developed for course assignments and the parameters were set by manual fine-tuning. In addition, I have validated the results for Neural Networks with Matlab's Neural Network Toolbox [18].

In addition, I chose the 3-fold cross validation where 2/3 flows were left for training and 1/3 for validation. Also, in the result section I have shown the effect of choosing various fractions (1%, 10%, 25%, 50%, 75% and 100%) of the training set on the performance metrics. Also, each of the experiments was run 200 times and the results are the average of the 200 executions. This was done to remove the effect of leaving out important training data in random selections.

4 Experiment and Results

In this section, I present the experiment steps and results obtained by running the classification algorithms on real data traffic.

Table 1: Uncertainty levels for Naive Bayes

Training Size	True Positive	False Positive	True Negative	False Negative
20	0.1995	0.0981	0.5159	0.1865
200	0.2401	0.0798	0.5343	0.1458
500	0.2501	0.0841	0.5299	0.1359
1000	0.2492	0.0868	0.5273	0.1367
1500	0.2485	0.0870	0.5271	0.1375
2000	0.2482	0.0872	0.5269	0.1377

4.1 Performance Metrics

I gathered the true positive, true negative, false positive and false negative rates and I used accuracy and precision metrics to measure the performance of Naive Bayes and Neural Networks.

- *Accuracy* of an algorithm is the ratio of sum of True Positives and True Negatives over the sum of True Positives, False Positives, True Negatives and False Negatives or the proportion of flows that are properly classified.
- *Precision* of an algorithm is the ratio of True Positives over the sum of True Positives and False Positives or the proportion of flows that are properly attributed to a given application (Skype) by this algorithm.

4.2 Capture and Extraction of the Flows Features

In order to compare the classification algorithms, I used the tcpdump tool available with the Linux Operating System to capture the raw packets. At the same time, I developed a script that probes netstat to gather the list of applications and their corresponding open port numbers. I saved the raw packets in the pcap format and the application name and port values in a format readable by Python programming language (pickle).

At the next step, I used Python and the Scapy library in Python to parse the raw packets and create a list of flows which satisfy the conditions mentioned in 3.1. Then, I used the netstat information to label each of the flows in the flow list. Moreover, I extracted the features for each of the flows in the list using the raw packet information and generated an output readable by Matlab.

Finally, I used the Matlab files to train my classifiers and gather the results.

4.3 Dataset

We have performed the learning on the following dataset:

- UBUNTU: refers to 24 hour long network trace collected on a Dell Optiplex 360 desktop running Ubuntu Operating System on the University of Toronto's Syslab network.

The UBUNTU dataset is representative of a typical data connection from a Linux machine to the Internet, in which there is a combination of TCP and UDP flows carrying web, email, ssh, Skype and file transfer services. Note that many common types of traffic such as P2P file sharing is missing from this traffic due to strict network prohibitions and location of traffic capture. Also, the Skype traffic includes Skype to Skype (E2E) audio call, text chatting and Skype to PSTN (SkypeOut) calls. The size of the captured traffic is 350 MB and the total number of flows (with at least 4 packets) for this trace is 3252 where 1241 of them are Skype flows. Without loss of generality, I have chosen the first 3000 flows (with 1156 Skype flows) for our experiment because it would be easier to split it for training and test.

4.4 Results

In this section we analyze the results from out experiment. The size of the test set for these experiments is fixed and equals to 1000 while the training size changes.

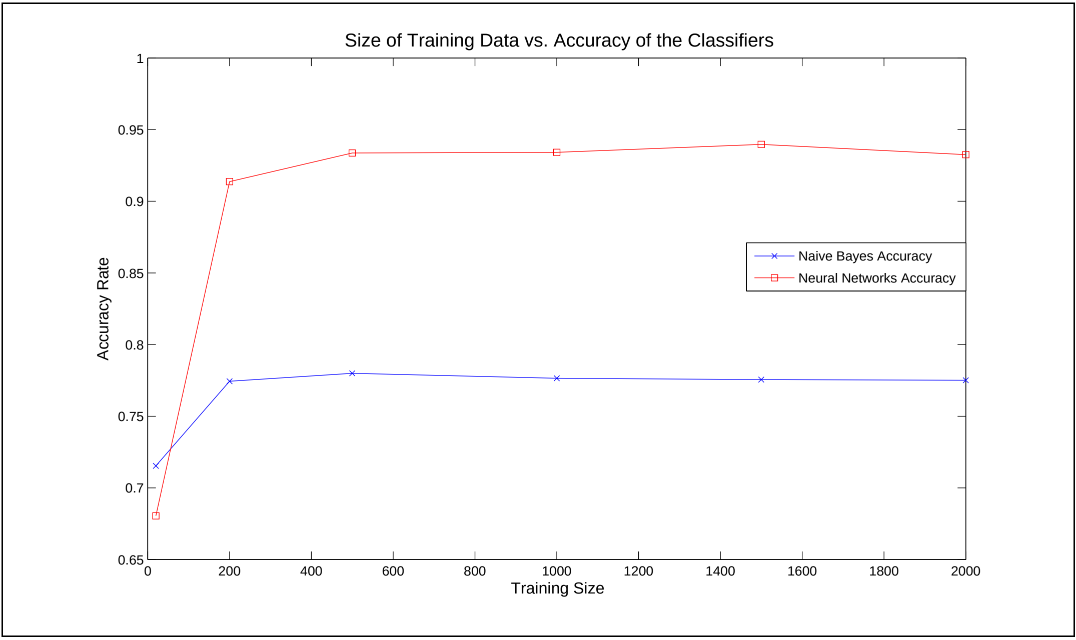

Figure 1: Accuracy of Naive Bayes and Neural Networks.

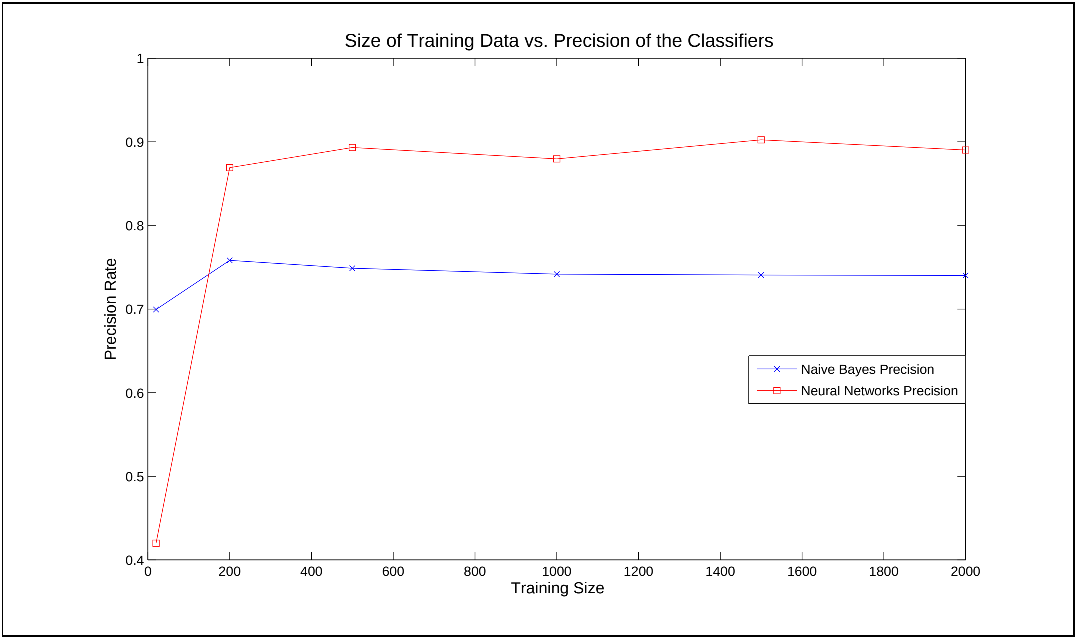

Figure 2: Precision of Naive Bayes and Neural Networks.

From figure 1, we can see that Neural Networks (NN) has a large advantage over Naive Bayes (NB) in both accuracy and precision except for when the training set is very small. This matches our expectation and the results from the previous research. Also, we can see that the amount of improvement for both of the algorithms is quite small when our training set becomes larger than 500 (50% of the size of the test set). This result can be very beneficial because the classifier can save time by training on a much smaller training set for NN. Moreover, the accuracy of NB matches the accuracy suggested by [2] ($\sim 78\%$) and the accuracy of NN matches the accuracy suggested by

Table 2: Uncertainty levels for Neural Network

Training Size	**True Positive**	**False Positive**	**True Negative**	**False Negative**
20	0.1871	0.1267	0.4919	0.1942
200	0.3523	0.0570	0.5617	0.0290
500	0.3604	0.0469	0.5718	0.0209
1000	0.3681	0.0617	0.5571	0.0132
1500	0.3632	0.0482	0.5705	0.0181
2000	0.3602	0.0490	0.5697	0.0212

Table 3: Execution times for Naive Bayes and Neural Networks in seconds

Training Size	**Naive Bayes**	**Neural Networks**
20	0.0009	0.8403
200	0.0010	1.5021
500	0.0011	2.6161
1000	0.0011	4.5059
1500	0.0014	6.3676
2000	0.0019	8.1971

[11] ($\sim 92\%$) which shows that by choosing appropriate features we can even accurately classify encrypted network traffic at a close accuracy rate to the non-encrypted traffic.

In addition, from figure 2, we can see that for both of these algorithms, the changes in the precision and accuracy are very similar. This shows that the features chosen works well for both Skype and non-Skype traffic.

Moreover, from table 1 and 2, we can see that NB has a smaller false positive rate than NN when the training set is small (size = 20) and this value reaches its minimum around training size 200 and as the size increases, the true negative value decreses while false positive increases. This can be because of overfitting. A similar situation occurs for NN when the training size grows larger than 500.

Finally, from table 3, we can see that the execution time for NN increases substantially as the training size grows (in the order of few seconds) while NB's execution time does not change significantly. Also, we know that for a real-time online traffic classifier the classification time is very important because most of the flows last less than a second and if the classifier responds after the flow termination then the classification will be useless. Therefore, I would recommend using NB approach in this scenario, even though it has a significantly lower accuracy and precision.

5 Conclusion

I conducted a detailed comparison of Naive Bayes and Neural Networks approaches for Skype traffic classification. My study confirmed the previous research and also yielded several insights: (a) Neural Networks is significantly more accurate than Naive Bayes for classifying Skype Traffic and it is definitely the algorithm of choice for offline traffic classification. (b) Both of the approaches need a training set smaller than half of the test set to reach very accurate results. (c) Neural Networks takes a long time to execute and may not be effective for online traffic classification. Therefore, Naive Bayes would be more beneficial in real-time traffic classifiers.

Scientifically grounded traffic classification research requires that researchers share tools and algorithms, and baseline data sets from a wide range of Internet links to reproduce results. In pursuit of this goal, I will make my code, dataset labeled with ground truth, and classifiers available for researchers interested in validating or extending my work.

6 References

[1] Skype web site, http://www.skype.com

[2] D. Bonfiglio, M. Mellia, M. Meo, D. Rossi, and P. Tofanelli, Revealing skype traffic: when randomness plays with you, ACM SIGCOMM Computer Communication Review, vol. 37, 2007, p. 48.

[3] A. Buonerba, Skype Traffic Detection and Characterization, HELSINKI UNIVERSITY OF TECHNOLOGY, 2007.

[4] D. Adami, C. Callegari, S. Giordano, M. Pagano, and T. Pepe, A Real-Time Algorithm for Skype Traffic Detection and Classification, Proceedings of the 9th International Conference on Smart Spaces and Next Generation Wired/Wireless Networking and Second Conference on Smart Spaces, 2009, p. 179.

[5] S.A. Baset and H. Schulzrinne, An analysis of the skype peer-to-peer internet telephony protocol, IEEE infocom, 2006.

[6] S. Ehlert, S. Petgang, T. Magedanz, and D. Sisalem, Analysis and signature of Skype VoIP session traffic, CIIT 2006: 4th IASTED International Conference on Communications, Internet, and Information Technology, 2006, pp. 8389.

[7] F.I. Khan, A Generic Technique for Voice over Internet Protocol (VoIP) Traffic Detection, IJCSNS, vol. 8, 2008, p. 52.

[8] J.P. Curtis, J.G. Cleary, A.J. McGregor, M.W. Pearson, Measurement of Voice Over IP Traffic, Proceedings of PAM-2000: April 2000, Hamilton, New Zealand.

[9] A.W. Moore and D. Zuev, Internet traffic classification using bayesian analysis techniques, ACM SIGMETRICS Performance Evaluation Review, vol. 33, 2005, pp. 5060.

[10] [1] T. Karagiannis, K. Papagiannaki, and M. Faloutsos, BLINC: multilevel traffic classification in the dark, Proceedings of the 2005 conference on Applications, technologies, architectures, and protocols for computer communications, 2005, p. 240.

[11] H. Kim, K.C. Claffy, M. Fomenkov, D. Barman, M. Faloutsos, and K.Y. Lee, Internet traffic classification demystified: Myths, caveats, and the best practices, Proceedings of the 2008 ACM CoNEXT conference, 2008.

[12] CoralReef. http://www.caida.org/tools/measurement/coralreef/.

[13] Z. Li, R. Yuan, and X. Guan, Accurate classification of the internet traffic based on the SVM method, IEEE International Conference on Communications, 2007. ICC'07, 2007, pp. 13731378.

[14] Y. Zeng and T.M. Chen, Classification of Traffic Flows into QoS Classes by Unsupervised Learning and KNN Clustering, KSII Transactions on Internet and Information Systems (TIIS), vol. 3, 2009, pp. 134146.

[15] D. Barman and M. Faloutsos, Comparison of Internet Traffic Classification Tools.

[16] J. Erman, M. Arlitt, and A. Mahanti. Traffic Classificaton Using Clustering Algorithms. In ACM SIGCOMM MineNet Workshop, September 2006

[17] WEKA: Data Mining Software in Java. http://www.cs.waikato.ac.nz/ml/weka/.

[18] MATLAB - Neural Network Toolbox. http://www.mathworks.com/products/neuralnet/

[19] N. Williams, S. Zander, and G. Armitage. A preliminary performance comparison of five machine learning algorithms for practical ip traffic flow classification. ACM SIGCOMM CCR, 36(5):715, October 2006.

[20] T. Auld, A. W. Moore, and S. F. Gull. Bayesian neural networks for internet traffic classification. IEEE Transactions on Neural Networks, 18(1):223239, January 2007.

[21] N. Williams, S. Zander, and G. Armitage. Evaluating machine learning algorithms for automated network application identification. Technical Report 060401B, CAIA, Swinburne Univ., April 2006.