Towards User Recognition by Shallow Web Traffic Inspection*

Marino Miculan, Gian Luca Foresti, and Claudio Piciarelli

Department of Mathematics, Computer Science and Physics, University of Udine
marino.miculan@uniud.it, gianluca.foresti@uniud.it,
claudio.piciarelli@uniud.it

Abstract. We consider the problem of *web user recognition*, or *web traffic de-anonymization*: given a set of users, is it possible to identify which user has generated a given web traffic with only a shallow packet inspection (that is, without looking inside the packet payloads)?
We propose to address this problem by means of *machine learning* (ML) techniques, in particular *clustering* and *supervised classification*. The basic idea is that each user can be identified by their browsing habits, and these habits can be described by a suitable set of features: click frequency, permanence time on web pages, amount of downloaded data, etc. In this paper we introduce these features, and show how these can be derived from the data obtained only from packet headers and their arrival time. Finally, we show the effectiveness of this approach with some preliminary tests and experiments.

1 Introduction

Nowadays, it is important to be able to identify the users accessing the Internet both for commercial and forensics purposes. On one side, Internet Service Providers and Content Providers can take advantage of this analysis, in order to personalize and improve their services; on the other, it is important to identify users in order to ascertain the responsibility for harmful or even criminal actions.

In fact, ISPs are already required to keep *access log files*, whose acquisition from the authorities is regulated by Directives such as 2006-24-CE, both for cybercrimes against persons (such as online fraud and identity theft) and for attacks against a server or a network. However, these access logs record only limited information, such as connection times, transfer amounts, and visited web site addresses, which in general are not sufficient to trace the specific author of an offense, but only the holder of the network connection contract. This is the case of networks which access the Internet via one or few public IP addresses by means of the well-known *network address translation* technique, such as in the case of home networks, small enterprises, commercial premises offering WiFi connection to their customers, etc.. Another scenario is that of workstations accessible by many users in public or semi-public environments, like college laboratories, hotel

* Partially supported by UniUd PRID 2017 *ENCASE* and by Italy-Singapore bilateral technology cooperation project PRESNET.

lobbies, internet cafés, etc.. In these cases, we cannot associate the traffic from a given IP address to a specific user, because a single public IP address is shared among several users, possibly at the same time: family members and their visiting friends, employees and their collaborators, students, hotel guests, customers, etc..

One could argue that more useful information can be obtained by looking for relevant data (e.g. usernames, email addresses) inside the payloads of IP packets. This technique, called *deep packet inspection* (DPI), is well-known and can be very effective [5, 8], but it can be applied only if the traffic is not encrypted. Nowadays most web traffic (especially that carrying identification data) is encrypted at the transport level by means of SSL and TLS protocols; actually an increasing number of websites is adopting encryption protocols, and therefore DPI will be less and less applicable. Moreover, DPI raises important privacy issues, because it allows the inspector to access the whole traffic content, not only the data needed for user identification [2].

Therefore, the problem is: given a set of users, is it possible to identify which user has generated a given web traffic, by means of *shallow* packet inspection? By "shallow" we mean that the only web traffic data we are allowed to consider are those an ISP can normally access for providing its service: the network (IP) header, the transport (TCP/UDP) header, the sizes and frequency of packets, etc., but not the content of the TCP payload. Shallow packet inspection is a novel technique that only very recently has been investigated by the research community, not for de-anonymization but for traffic classification [12, 6, 11].

In this paper, we propose to address this problem by means of *machine learning* (ML) techniques, in particular *clustering* and *supervised classification*. The basic idea is that each user can be identified by their browsing habits, and that these habits can be described by a suitable set of features, such as traffic size, click frequency, permanence time on web pages, hour of the day of the activity, etc. In this paper we introduce some of these features, and show how these can be derived from the data obtained by shallow packet inspection. To this end, we first introduce the notion of *click* as the basic event to consider in the analysis. A "click" represents the voluntary action performed by the user when clicking on a hyperlink/button on a web page. Therefore, a click subsumes the whole traffic (that is, all the HTTP(S) requests and replies) caused by this action. Click data are obtained by partitioning the packet flow to/from the observed IP address, using clustering techniques. Then, on the flow of clicks we identify suitable features for the classification. Some features are intuitive (e.g. counterpart IP address, traffic size), but others are less obvious, such as the time of the day when the click is performed, or the "dwell time" on a web page. These data are related to the browsing habits of each user, and hence can be used as the basis for the supervised classification algorithms. We consider different algorithms; each of them is trained and tested with the same sets of click flows. The results are encouraging: some algorithms yield high classification precision.

The rest of the paper is organized as follows. In Section 2 we present in detail the problem under consideration. The solution we propose is described in Section 3, and experimental results are reported in Section 4. Finally in Section 5 we draw some conclusions and outline future work.

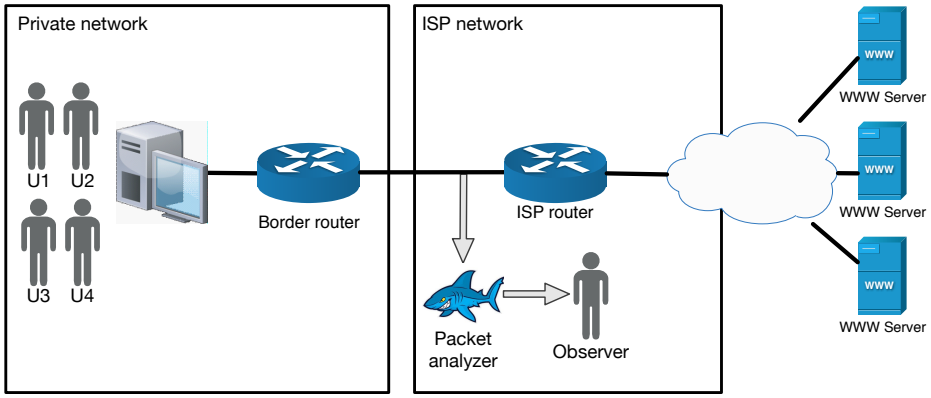

Fig. 1. Example scenario.

2 Problem description and analysis

Let us consider the scenario shown in Figure 1. In this scenario, there is a private network where several users $u_1, \ldots, u_n$ can browse the Web using the same client computer. During a session, the client computer is used by only one user. The browsing activity of the user generates a sequence of HTTP(S) requests towards various WWW servers on the Internet, and corresponding replies. In turn, these requests yield a flow of IP packets from the client to the web servers and back. These flows go through the border router and the ISP router(s), where can be examined by an observer. Very often, the private network uses a non-routable address space (such as 10.0.0.0/8 or 192.168.0.0/16). In this case the border router performs a NAT/PAT translation [10]: in each outgoing packet the source addresses and port are replaced with the public IP address assigned to the border router by the ISP—and dually for the incoming packets. The net effect is that, from the public network viewpoint, the whole private network appears as a single host with the public IP address.

Using a packet analyzer, the observer can gather the following data for each packet:

- Arrival time at the router;
- Total length of the packet;
- Source IP address and port number: for packets coming from the border router, the source IP address is the public IP assigned to the local router by the ISP, and the port is a dynamic port on such router. For packets going to the border router, the source IP address and port number are those of the contacted web server;
- Destination IP address and port number: for packets coming from the local router, the destination IP address and port number are those of the contacted web server; for packets going to the local router, the destination IP address is the public IP assigned to the local router by the ISP, and the port is a dynamic port on such router.

Other data in the headers can be ignored because either not relevant (such as TOS, or fragmentation details), or non informative (the version of IP is almost always 4, the trasport level protocol is always TCP, etc.). It is important to notice that these data cannot be encrypted, otherwise the routers would not be able to route the packet. Following the shallow packet inspection policy, we do not analyze the content of TCP segments; very likely these segments carry SSL/TLS encrypted payloads, which cannot be analyzed further.

Therefore, for each web session we can obtain a file, called *web session log*, of tuples of the following form:

$\langle$*arrival time*, *packet length*, *source IP*, *source port*, *destination IP*, *destination port*$\rangle$.

Since we intend to apply supervised classification algorithms, we assume to be able to obtain a suitable *training set* in order to "learn" the browsing habits of each user. A training set is a set $TS = \{\langle L_1, u_{i_1}\rangle, \ldots, \langle L_k, u_{i_k}\rangle\}$ of web session logs associated to the corresponding user. Here, i_k is the index of the user generating log L_k. This set can be build by observing the traffic when we know the actual identity of the user browsing at that moment.

Then, the classification problem can be formulated as follows:

> given a training set TS for users $u_1, \ldots u_n$ and a web session log L generated by one of these users, is it possible to determine which user has generated L?

The criteria for evaluating a classifier are the usual ones:

Accuracy: the percentage of web session logs correctly classified;
Recall: the percentage of web session logs correctly assigned to a user, with respect to all logs generated by that user;
Precision: the percentage of web session logs correctly assigned to a user, with respect to all logs assigned to that user;
F-measure: the harmonic average of recall and precision.

In the next section we propose a machine learning-based solution to this problem.

3 Solution proposal

In order to recognize users by means of shallow packet inspection, we propose the architecture depicted in Figure 2. When building the training set, we assume that users can be identified by the IP address of their workstation, thus no user can access to more than one workstation, and no workstation can be used by more than one user. Moreover, we assume that no NAT policy is implemented. These requirements are only needed to support the training phase of a supervised classifier, where each data sample must be labeled with the correct classification; user identification is not needed in classification phase, except for performance measurement. The whole network traffic is logged by a sniffer and subsequently filtered and pre-processed to collect only the data relevant for the

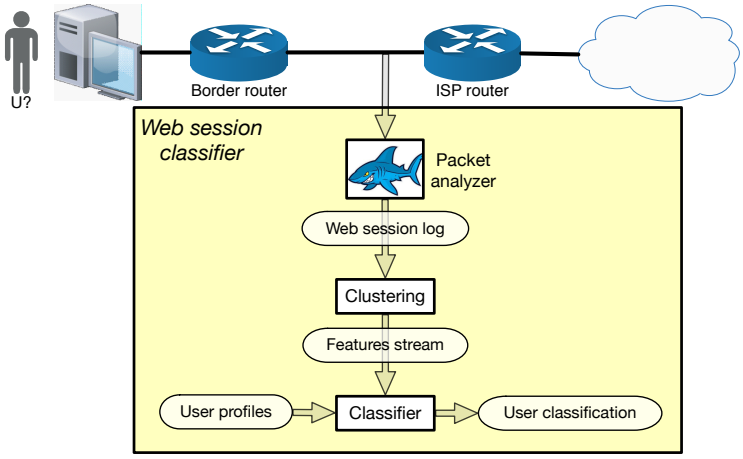

Fig. 2. Architecture of the web session classifier.

system. Pre-processing also includes a clustering step, in which data associated to the same user action are grouped to identify meaningful high-level data that are fed to the classifier for training or evaluation. In order to preserve user privacy, pre-processing also replaces source IP addresses with unique identifiers (User1, User2, etc.). Hence, despite the system internally stores the address/identifier associations (which are needed to guarantee a coherent labeling through time), the final data are pseudonymized and can be safely shared.

3.1 Feature extraction and filtering

The sniffer acquires and logs all the network traffic coming to/from the network; it is thus placed either in the local network itself (where it can access the data by enabling the promiscuous mode of the network card) or in a bottleneck device such a network router. In case of small networks, it can be implemented using publicly available software, such as WireShark. However, in most cases a full log of network traffic quickly leads to extremely large log files, which are both impractical to store and pose privacy issues, since they may contain data outside the scope of the proposed system. We thus adopt filtering rules on the sniffer, in order to log only relevant traffic. In particular, we assume that most user actions generate TCP/IP traffic, thus any other packet type is silently ignored. This include both user-generated data, such as UDP/IP packets, as well as network management data, such as ARP, SNMP, RIP packets etc.. Moreover, in this work we choose to focus only on a specific type of data, namely web traffic, as the Web is a popular service that may contain several user-distinctive features to ease the classification task. The sniffer is thus configured to acquire only TCP traffic to or from port 80 (HTTP) or 443 (HTTPS). The encryption of HTTPS connections is not an issue, since the proposed system performs a shallow inspection, without analyzing the packet payload. Finally, to further reduce the amount of data to

be stored, we extract only relevant features form each packet. As mentioned in Section 2, the final collected raw data are:

- packet arrival time;
- client IP address and TCP port;
- server IP address and TCP port;
- packet length.

The client IP address is then pseudonymized as described at the beginning of this section.

3.2 Feature pre-processing

Since the goal of the proposed system is to identify users by their web navigation behaviors, it is important that data can be associated to voluntary user actions. This task is not trivial since network traffic generated by modern web navigation can be loosely connected to user actions. Let us consider for example the basic action of clicking on a hyperlink. The corresponding network traffic delivers the required web page, but also new, parallel connections deliver other page contents (e.g. images, cookies, etc.). Moreover, new connections can be established to other web servers, such as advertisement-delivery services, profiling services etc.

Thus, in order to work on a higher abstraction level we introduce the notion of *click*, defined as the set of all network traffic generated by a user action, such as clicking on a hyperlink. Clicks can be extracted by temporally clustering the data packets: groups of data packets with close arrival times are considered part of the same click, even if originating from different servers.

Since the number of expected clicks is unknown *a priori*, no algorithms requiring an initial knowledge on the number of clusters is suitable, thus excluding popular clustering techniques such as k-means or Gaussian Mixture Models. Moreover, we require hard clustering (cluster membership is a binary choice) and explicit outlier modeling, since not all the network traffic could belong to a click. These considerations motivate the choice of DBSCAN [3] as clustering algorithm. DBSCAN uses a density-based approach, where clusters are defined as groups of high-density samples. Formally, given a set of samples P, we give the following definitions:

- $p \in P$ is a *core point* if at least m points $q_1 \dots q_m \in P$ exist such that $\|P - q_i\| \leq \epsilon \; \forall i \in [1 \dots m]$, where m, ϵ are the algorithm parameters;
- $q \in P$ is *directly-reachable* from $p \in P$ if $\|p - q\| \leq \epsilon$ and p is a core point;
- $q \in P$ is *density-reachable* from $p \in P$ if there exists a path $p_1 \dots p_n$ such that $p_1 = p, p_n = q$ and p_{i+1} is directly-reachable from $p_i \; \forall i \in [1 \dots n-1]$.

Given a core point p, its cluster is then defined as the set of all the points that are density-reachable from p. Points that are not density-reachable by any other point are marked as outliers. Figure 3 shows the effect of DBSCAN applied to the arrival time of a set of packets in order to identify clicks and outliers.

For each click, we compute the following features:

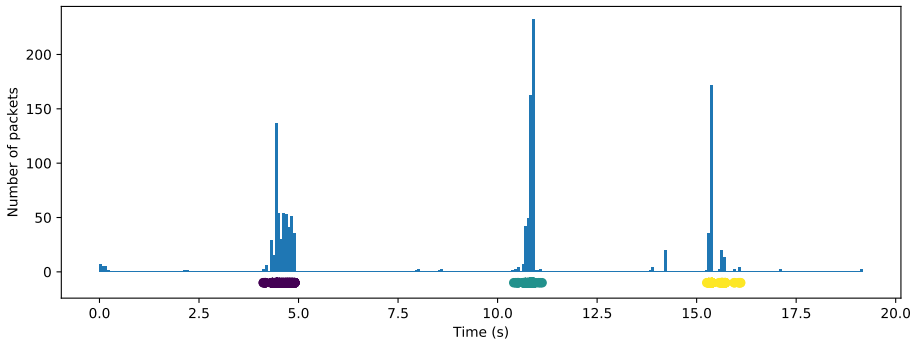

Fig. 3. Histogram of the number of detected packets while clicking on three links. Colored areas under the histogram show the three clicks detected by the proposed clustering procedure.

- user ID;
- main site, i.e. the IP address of the destination server of the first packet in the click;
- timestamp, defined as the arrival time of the first packet in the click;
- inter-click time, defined as the time passed since the last click from the same user to the same main site (set to 0 if it is the first one);
- total amount of data, defined as the sum of all packet lengths in the click;
- total number of secondary sites, i.e. destination servers different from the main site.

Furthermore, by analyzing all the clicks originated from the same user, we define a *session log* as a set of statistics about the acquired clicks. A session log is thus a data sample containing:

- user ID;
- main site;
- session start time, defined as the timestamp of the first click;
- session end time, defined as the timestamp of the last click;
- session duration, defined as the the difference between session end time and start time;
- total number of clicks in the session;
- average inter-click time, defined as session duration / total number of clicks;
- average click data length, defined as total amount of data / total number of clicks;
- average number of secondary sites.

As a final pre-processing step, all the numerical data are standardized, since this is required by many machine learning algorithms. Standardization is achieved by mean removal and variance scaling: given a set of feature values $\{f_1 \dots f_n\}$,

each feature value f_i is replaced by its standardized version $\hat{f}_i$ defined as:

$$\hat{f}_i = \frac{f_i - \bar{f}}{\sqrt{1/n \sum_{j=1}^{n} (f_j - \bar{f})^2}} \tag{1}$$

where $\bar{f} = 1/n \sum_{j=1}^{n} f_j$. Feature means and standard deviations are computed on the training set only. Test sets are standardized using the same values.

3.3 Classification algorithms

The acquired session data are used to train a machine learning classifier. As features, we consider all the numerical data of a session. The main site, despite being a strong hint for user identification, is currently discarded since its categorical, rather than numerical, nature poses extra processing difficulties that will be addressed in a future work. The User IDs of each session are used as sample labels.

In order to classify the data, we considered the following algorithms:

Naive Bayes Naive Bayes classifiers [13] define $P(y|x_1 \dots x_n)$ as the probability of class y given the features $x_1 \dots x_n$. Under the naive (hence the name) assumption of conditional independence between every pair of features given the value of the class variable, it can be proven that:

$$P(y|x_1 \dots x_n) \propto P(y) \prod_{i=1}^{n} P(x_i|y) \tag{2}$$

and the class estimate $\hat{y}$ is thus defined as:

$$\hat{y} = \arg\max_{y} P(y) \prod_{y=1}^{n} P(x_i|y) \tag{3}$$

where $P(y)$ and $P(x_i|y)$ can be estimated from data using maximum a posteriori (MAP) estimation. In this work we adopted a Gaussian Naive Bayes model, where $P(x_i|y)$ is assumed to be a Gaussian function.

K-Star The K-Star classifier [7], or K^*, is an instance-based classifier, meaning that the class of an instance is based upon the class of those training instances similar to it, as determined by some similarity function. Specifically, K-Star adopts a entropy-based distance function, calculated by mean of the complexity of transforming an instance into another.

Support Vector Machines Linear Support Vector Machines [1] are based on the idea that an optimal linear classifier maximizes the margin, this is the width of the strip parallel to the classification hyperplane that separates the two classes.

The solution is found by solving a constrained optimization model leading to the following classification function:

$$f(x) = \text{sgn}\left(\sum_{i=1}^{n} y_i \alpha_i (x \cdot x_i) + b\right) \tag{4}$$

where x_i is a sample from the training set and y_i is the corresponding class label, while α and b are found by solving the optimization problem. The solution is actually sparse since $\alpha_i = 0$ for most of the data, except the few ones lying on the margin (support vectors). SVMs became popular because they can be easily extended to the non-linear case by means of kernel methods.

C4.5 The C4.5 algorithm [9] is a decision tree building algorithm based on its precursor ID3. The decision tree is built using the concept of information entropy: at each node, C4.5 chooses the feature that most effectively splits the data associated to the branch. The effectiveness of the split is measured in terms of information gain (difference in entropy) of the feature. Once built, the decision tree can be easily used to classify new data.

4 Tests and evaluation

In oder to evaluate the system performances, we tested the system on a Local Area Network where 10 users (both male and female, in the age range of 20–45 years) were asked to visit web pages as in their normal daily routine. We did not consider the case of a malicious user deliberately trying to hide their network traffic pattern. Users were informed about privacy issues, to clarify that only web traffic metadata was logged and no deep inspection was performed. This way, users felt free to use the web as in their normal activity. Data were acquired along 10 sessions, each one 10 minutes long. On average, we collected ~ 300 clicks per user. The relatively small amount of data motivates the choice of the algorithms presented in Section 3, since more sophisticated techniques such as deep neural networks would require lager datasets.

After pre-processing, the dataset was split in a training and a test set with different ratios to evaluate the performances on low amounts of training data. Tests were performed using respectively 20%, 50% and 80% of the original data as training set. Results obtained with the four classifiers are shown in Table 1.

As it can be seen, all the classifiers achieved good performances except Support Vector Machines. This could be explained by a poor choice of training parameters, since SVM results heavily depends on the choice of kernel and kernel parameters. As a future work, we plan to further investigate this aspect. Among the remaninig classifiers, C4.5 performed best, reaching high accuracy levels even with a small training set (20% of total data). The preliminary results are thus encouraging, and in the next future we aim to test the system with a larger user base.

Classifier	Accuracy (%)	Precision (%)	Recall (%)	F-Score (%)
Naive Bayes 20%	**95**	96	95	95
Naive Bayes 50%	**97**	98	97	97
Naive Bayes 80%	**96**	96	96	96
K-Star 20%	**75**	76	75	75
K-Star 50%	**84**	84	84	84
K-Star 80%	**84**	86	84	85
SVM 20%	**44**	69	44	45
SVM 50%	**54**	69	53	55
SVM 80%	**59**	75	59	62
C4.5 20%	**97**	97	97	97
C4.5 50%	**99**	99	99	99
C4.5 80%	**99**	100	100	100

Table 1. Classification results.

5 Conclusions

The ability to identify web users by analyzing their network traffic can have multiple applications, from user profiling to digital forensics. In this paper we investigated the possibility of identifying users only by means of shallow inspection of HTTP(S) network traffic. Shallow inspection, in which the content of the packet payload is not analyzed, is motivated both by privacy issues and by technological factors: nowadays, the increasing adoption of encrypted connections is making deep inspection mostly useless. Despite the few amount of data gathered by shallow inspection, we proposed a data pre-processing method to extract high-level features that could be relevant for user identification, such as inter-click time intervals, time spent on a single web page, etc.. We tested four different classifiers on a small dataset obtaining encouraging preliminary results.

As a future work, we plan to acquire a larger dataset in order to test more complex classifiers such as deep neural networks. Moreover, we intend to investigate the reasons for the relatively low performance of the SVM classifier, in particular concerning the choice of the kernel and kernel parameters. We also plan to evaluate if automatic deep feature extraction techniques can actually outperform our manually-defined high-level feature set [4]. Finally, we will also focus on proper representation and processing of categorical data, in order to handle non-numerical features such as server IP addresses.

Acknowledgments We thank Clelia Bincoletto for preliminary work and experiments on the subject of this paper.

References

1. Cristianini, N., Shawe-Taylor, J., et al.: An introduction to support vector machines and other kernel-based learning methods. Cambridge university press (2000)
2. Daly, A.: The legality of deep packet inspection. International Journal of Communications Law & Policy (2011). https://doi.org/10.2139/ssrn.1628024

3. Ester, M., Kriegel, H.P., Sander, J., Xu, X., et al.: A density-based algorithm for discovering clusters in large spatial databases with noise. In: Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96). vol. 96, pp. 226–231 (1996)
4. Goodfellow, I., Bengio, Y., Courville, A., Bach, F.: Deep learning. MIT press (2016)
5. Kumar, S., Turner, J., Williams, J.: Advanced algorithms for fast and scalable deep packet inspection. In: Architecture for Networking and Communications systems, 2006. ANCS 2006. ACM/IEEE Symposium on. pp. 81–92. IEEE (2006)
6. Lotfollahi, M., Shirali, R., Siavoshani, M.J., Saberian, M.: Deep packet: A novel approach for encrypted traffic classification using deep learning. arXiv preprint arXiv:1709.02656 (2017)
7. Painuli, S., Elangovan, M., Sugumaran, V.: Tool condition monitoring using k-star algorithm. Expert Systems with Applications **41**(6), 2638–2643 (2014)
8. Parsons, C.: Deep Packet Inspection in Perspective: Tracing its lineage and surveillance potentials. Surveillance Studies Centre, Queen's University (2008)
9. Quinlan, J.R.: C4. 5: programs for machine learning. Elsevier (2014)
10. Srisuresh, P., Holdrege, M.: IP Network Address Translator (NAT) Terminology and Considerations. The Internet Society (1999), rFC 2663
11. Velea, R., Ciobanu, C., Gurzau, F., Patriciu, V.V.: Feature extraction and visualization for network pcapng traces. In: Control Systems and Computer Science (CSCS), 2017 21st International Conference on. pp. 311–316. IEEE (2017)
12. Velea, R., Ciobanu, C., Mărgărit, L., Bica, I.: Network traffic anomaly detection using shallow packet inspection and parallel k-means data clustering. Studies in Informatics and Control **26**(4), 387–396 (2017)
13. Zhang, H.: The optimality of naive Bayes. Proc. Seventeenth International Florida Artificial Intelligence Research Society Conference, FLAIRS 2004 (2004)