Knowledge-Based Systems 197 (2020) 105875

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

Identifying legitimate Web users and bots with different traffic profiles — an Information Bottleneck approach

Grażyna Suchacka *, Jacek Iwański

Institute of Informatics, University of Opole, Opole, Poland

ARTICLE INFO

Article history:
Received 12 September 2019
Received in revised form 31 March 2020
Accepted 3 April 2020
Available online 11 April 2020

Keywords:
Web user
Web bot
Internet robot
Bot detection
Machine learning
Unsupervised learning

ABSTRACT

Recent studies reported that about half of Web users nowadays are intelligent agents (Web bots). Many bots are impersonators operating at a very high sophistication level, trying to emulate navigational behaviors of legitimate users (humans). Moreover, bot technology continues to evolve which makes bot detection even harder. To deal with this problem, many advanced methods for differentiating bots from humans have been proposed, a large part of which relies on supervised machine learning techniques. In this paper, we propose a novel approach to identify various profiles of bots and humans which combines feature selection and unsupervised learning of HTTP-level traffic patterns to develop a user session classification model. Session clustering is performed with the agglomerative Information Bottleneck (aIB) algorithm, as well as with some other reference algorithms. The model is then used to classify new sessions to one of the profiles and to label the sessions as performed by bots or humans. An extensive experimental study, based on real server log data, demonstrates the ability of aIB clustering to distinguish user profiles and confirms high performance of the classification model in terms of accuracy, F1, recall, and precision.

1. Introduction

Expansion of modern Web-based services, such as social media or electronic business, has gone hand in hand with development of autonomous intelligent agents, known as Internet robots, Web bots, or crawlers. Robots are software tools which make use of the hyperlink-based structure of the Web to access Web pages and use the collected information to perform various tasks. Bots are commonly employed to perform the work of indexing, updating, and maintaining the Web content, with such a salient example being search engine crawlers. On the other hand, however, bots are increasingly exploited for malicious and illegal purposes, like launching Denial of Service (DoS) attacks, collecting and reusing sensitive information, or taking an undue advantage in e-business in a lot of ways.

Recent industry reports reveal that about half of all Web users nowadays are intelligent agents, more than half of which are potentially harmful, "bad" bots [1,2]. Furthermore, the sophistication level of bad bots is increasing. According to [1], only about a quarter of bad bots remain simple agents that connect to websites using automated scripts and may be easily detected based on their IP addresses or HTTP user agent fields. The remaining three-quarters are Advanced Persistent Bots (APBs), able to change IP addresses and user agents: they may operate at a moderate sophistication level, using "headless browser" software simulating real browser technology, or may be very sophisticated tools, interacting with sites via browser automation software or malware installed within real browsers. In addition to impersonating legitimate users, many APBs are able to efficiently imitate mouse movements and clicks typical for human online navigation and thus, may be very hard to detect.

There is an increasing need to develop advanced methods for identifying Web bots effectively. In this paper, we propose a novel approach to Web bot detection, named IBBI (Information Bottleneck approach for web Bot Identification). The approach is based on employing unsupervised machine learning (ML) to discover various profiles of users in the context of separating bots from humans. It relies on the Fisher Score algorithm for feature selection and the Information Bottleneck method for session clustering.

Application of *unsupervised* learning is motivated by the fact that in reality many bots are camouflaged and when we have a historical user session data set, some bot sessions may be improperly labeled as human-generated. This implies that some bots in a training data set used in an experimental study will remain unidentified and outcomes of a supervised learning approach to bot detection will be negatively affected by using mislabeled samples to train a ML model. Another reason for using unsupervised learning results from previous studies' findings that bots exhibit a wide range of navigation strategies [3,4]. Development of an

* Corresponding author.
E-mail addresses: gsuchacka@uni.opole.pl (G. Suchacka), jaceki@uni.opole.pl (J. Iwański).

https://doi.org/10.1016/j.knosys.2020.105875

unsupervised ML model makes it possible to discover multiple hidden patterns in robot sessions (as well as in the human ones).

The major contributions of our study are as follows:

1. We develop a user session model representing various types of robot and human sessions. The Fisher Score algorithm is applied to rank session features according to their ability to differentiate bots from humans and the subset of the most relevant features is selected experimentally.
2. We apply an agglomerative Information Bottleneck (aIB) algorithm to discover different user profiles based on hidden Web traffic patterns intrinsic to robot and human sessions.
3. The user session model, developed on a training session data set, is used in a novel bot detection approach, IBBI, to classify new, unseen sessions from a test data set to one of the previously identified profiles and to label the sessions as performed by bots or humans.
4. We perform an extensive experimental study on real e-commerce data to verify the ability of the proposed solution to distinguish human and robot profiles, as well as its robustness to classify new sessions. To tune IBBI parameter values (the number of the most relevant session features, the number of clusters, and the minimum training set size), we propose using a clustering entropy measure. Moreover, we compare performance of aIB with other benchmarking clustering techniques.

To the best of our knowledge, this is the first study that takes multiple types of Web bots and legitimate users into consideration, applies unsupervised learning of a user type based on the most relevant session features, and accomplishes an in-depth analysis of differences in the discovered traffic patterns. None of previous works applied unsupervised learning to analyze traffic patterns of various types of humans and bots in detail.

The rest of the paper is organized as follows. Section 2 outlines the preliminaries on processing user sessions on a Web server and formulates the problem. Section 3 reviews related work on Web bot detection with special focus on solutions applying unsupervised learning techniques. Section 4 discusses the proposed research methodology, followed by a discussion of experimental results achieved in Section 5. The last Section concludes the paper and outlines our prospective work.

2. Problem formulation

Access to the Web content is realized according to the client–server network processing (Fig. 1). Communication between client and server software proceeds according to the application-layer protocol HTTP (HyperText Transfer Protocol), relying on the lower-level TCP/IP protocol suite [5]. A client is typically a Web browser, operated by a human user, who accesses subsequent pages of a given website; in that case for each single page the browser generates a sequence of HTTP requests: the first request for a page description file and the following requests for embedded objects (e.g., images). A client may be also an artificial agent (bot), whose interaction with the server is not restricted by the logical website structure but depends solely on the algorithm underlying the bot implementation.

The client sends consecutive HTTP requests to the server through the Internet; the server uses its resources to generate responses and sends them back to the client. A Web server system can process many requests from many clients concurrently by creating multiple processes or threads of execution in a single process. By default the server is blind to the client type (legitimate user or bot). Moreover, since HTTP is a stateless protocol, a user session on a given website has to be maintained using additional mechanisms, like cookies.

Each request arriving to the server is registered in a standard server access log file. Let us analyze an example server log entry, recorded for a single HTTP request in the *NCSA Combined* log format: *46.4.87.105 - - [11/May/2015:04:48:29 +0200] "GET /rob.txt HTTP/1.0" 200 24660 "-" "Googlebot-Image/1.0"*. The following fields may be distinguished for that request:

- *46.4.87.105*: the client IP address;
- "- -": two optional fields, here not given: a client identifier and a user identifier used for authentication;
- *11/May/2015:04:48:29 +0200*: data and time stamp of the request arrival;
- *GET*: the HTTP method (GET is the most common method, used to download a server resource);
- */rob.txt*: URI (Uniform Resource Identifier) of the requested resource;
- *HTTP/1.0*: a version of HTTP protocol used by the client;
- *200*: HTTP status code representing the result of the request processing at the server (200 means the successful request service);
- *24660*: volume of data (in bytes) sent to the client in response;
- "-": a referrer (here not given), which for the first request in user session is URL (Uniform Resource Locator) which linked the client to a given website and for consecutive requests is URL of the previous page;
- *Googlebot-Image/1.0*: user agent describing the client software (here one can see that the client was a Google bot for image indexing).

Since a server log has a standardized format, it is possible to pre-process log entries to extract individual request fields and use them to reconstruct user sessions. A *user session* is a sequence of HTTP requests sent by a client within a single visit on a website. Based on requests' fields, a session may be described with various statistical features, like the total number of requests, number of page requests, percentage of page requests, percentage of successfully processed requests, and many others.

The problem of identifying user profiles on a Web server may be formalized as follows. Given a user session described with some HTTP-level traffic features, *assign* the session to the most appropriate profile and *label* it as performed by a bot or human.

We address this problem with a machine learning (ML) solution combining feature selection and unsupervised learning. Performance of the proposed solution is evaluated experimentally on real server log data. The hypothesis behind our experiments is that representing user sessions with the most relevant features allows an ML method to discover hidden patterns in sessions of bots and humans of different types without the supervision of ground truth labels; the developed classification model may be then used to identify types of new sessions, in particular, to discriminate bots from humans.

3. Related work

3.1. Web bot detection approaches

Due to the increasing sophistication of Internet robots and resulting difficulties in their identification, many approaches to the problem of bot detection have been proposed in recent years. Many of them were devoted to specific types of bots on the Web, like spambots [6,7], botnets [8,9], phishing attacks [10], or shilling attacks [11].

In this paper we are interested in identifying bots operating at the application layer, which do not exhibit an aggressive accessing behavior but rather try to mimic legitimate Web traffic.

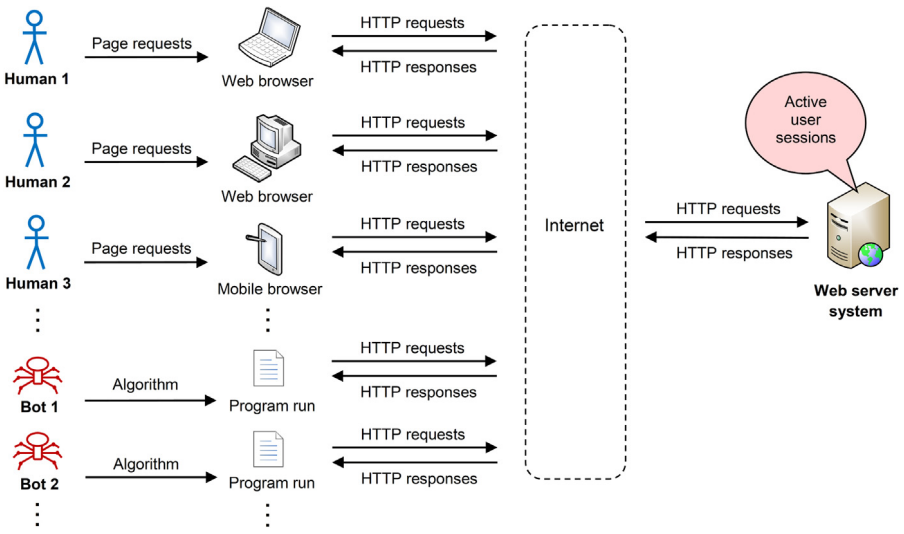

Fig. 1. Web service realized according to the HTTP protocol.

Previous studies investigated various Web traffic features representative for bots and humans. Early bot detection solutions were based on the analysis of known Web traffic patterns of both user types [12–14]. More advanced approaches investigated possibilities of analytical learning of hidden traffic patterns and developing a probabilistic or ML model [15–19].

Regarding application of ML techniques to bot detection, most approaches exploit supervised learning. The following methods have been investigated: decision trees, association rule mining, *k*-Nearest Neighbors, support vector machine, artificial neural network, Bayesian classifiers, and ensemble methods [3,13,20–28]. Most of these approaches address the problem of binary classification of user sessions (bots vs. humans).

The main drawback of bot detection solutions applying supervised learning is that their efficiency is highly dependent on the underlying session labeling strategy (the way of assigning ground truth labels to sessions used in the experimental analysis). Labeling procedures are fraught with an inherent error because bots are still evolving in terms of their sophistication and functions and thus, available expert knowledge-based databases of known bots' user agents and IP addresses, which are typically used in session labeling, are updated at a slower pace than the evolution of bots. This indicates the potential advantages of unsupervised learning solutions.

3.2. Unsupervised learning of web traffic patterns for bot detection

In recent years several methods were proposed to cluster user sessions in the context of distinguishing bots from legitimate users [3,4,24,29–31]. The efficiency of such approaches depends both on machine learning techniques applied and a set of session features used to describe user online behavior.

Stevanovic et al. [4] applied two unsupervised neural network learning algorithms to partition website visitors based on ten session features. The Modified Adaptive Resonance Theory 2 (Modified ART2) was used to generate five equal-size clusters and the Self-Organizing Map (SOM) was used to visualize the spacial distribution (proximity) of the clusters. The cluster contents was analyzed in terms of four user types: humans, well-behaved bots, malicious bots, and unknown visitors. As a result, two-dimensional SOM maps indicated a quite good separation between humans and bots although some users labeled as malicious bots revealed traffic patterns similar to these of humans. ART2 results showed that sessions of the four types are distributed across most of the generated clusters and no clear separation of client types was achieved. The general conclusion was that a large part of malicious bots reveals a very human-like behavior.

Zabihi et al. [31] applied Density-Based Spatial Clustering of Applications with Noises (DBSCAN) to assign users into two clusters: human- and bot-dominated, whereas some outlier sessions were treated as a "noise" and remained unclassified. A set of 14 session features was considered, from which four features were finally selected for clustering by the means of t-test. The clustering quality was assessed with supervised-oriented metrics without taking into account unclassified sessions — the mean entropy and purity scores were equal to 0.024 and 0.966, respectively. Similarly, Hamidzadeh et al. [29] combined an unsupervised learning approach with feature selection to partition Web bots and humans into clusters. Fuzzy Rough Set (FRS) theory was employed to overcome the curse of dimensionality and reduce an input 30-element feature set to 6–9 features and the SOM algorithm was used to cluster sessions. Clustering results were reported with entropy, purity, G-mean, and Jaccard measures, whose mean rates were equal to 0.37, 96.3%, 94.7%, and 88.3%, respectively. However, less than 60% of malicious bots on average were correctly assigned to bot-dominated clusters.

Both of the aforementioned studies concluded that augmenting unsupervised classification with a feature selection stage has a potential to improve classification performance rates. This was also confirmed in [30], where Principal Component Analysis (PCA) was applied to reduce the initial 40-feature set at the input of the *k*-means algorithm, aiming to separate bots from humans. The experimental analysis showed that either using top principal components as new, information-rich session features, or selecting a subset of the most significant features indicated by PCA, finally resulted in the increase of the clustering purity for ten clusters from 94% to 96%.

Lastly, very few studies addressed the problem of exploiting the unsupervised learning session model for binary classification (i.e., identification) of bot and human sessions [3,24,32]. In [32] a method called SMART (Soft computing for MAlicious RobotT detection) was proposed to classify humans and bots. The approach combined the Forward Approximation Algorithm (FAA) to select a feature subset from a 30-element set and the Markov clustering (MCL) algorithm. The developed model was then used to classify new sessions as generated by humans or bots, achieving recall of 0.47 and 0.93, as well as accuracy of 0.91 and 0.93, depending on the data set. Classification results were also evaluated

in terms of humans, benign bots, and malicious bots: as much as 89.6% and 84% of users classified to malicious bot clusters turned out to be correctly classified malicious bots. In [24] and [3] two unsupervised learning techniques were applied for micro-clustering of bots and humans on an e-commerce site: *k*-means and Graded Possibilistic *c*-Means (GPCM), based on 22 session features. The model was then used for binary classification of new user sessions, achieving both recall and accuracy rates at the level of 0.98.

The above analysis of the literature indicates that application of unsupervised learning techniques to identify Web robots is a relatively new yet a fast-developing research area. Furthermore, differences in Web traffic patterns have been analyzed only in terms of binary discrimination of humans from bots or have been based on a general yet unclear division of bots into benign and malicious ones. To the best of our knowledge, Web traffic patterns for multiple human and bot categories have not been investigated so far. We address this research issue by developing a classification model based on the unsupervised learning, verifying the model ability to identify new bot and human sessions on a website and analyzing the results taking various user types into account. Experiments are performed using a user session dataset reconstructed from HTTP data for a real e-commerce website.

4. Methodological framework of the proposed approach

We propose a machine learning approach IBBI (Information Bottleneck approach for web Bot Identification) to develop a user session classification model. The flowchart of the approach is presented in Fig. 2.

The input for IBBI consists of text files containing Web server log data in the *NCSA Combined* log format. The approach is based on a pipeline of data analysis to reconstruct user sessions, development of a feature vector-based session representation, ground truth session labeling, and training a session classification model through the unsupervised learning of traffic patterns. The model relies on the Fisher Score (FS) method for session feature selection and the agglomerative Information Bottleneck algorithm for session clustering. After learning the model on a training data set, it is used to classify new sessions from a test data set. In this section the proposed research methodology is discussed in detail.

4.1. Development of session representation

Log data pre-processing and reconstruction of user sessions was performed using a dedicated C++ log analyzer. Data streams from individual Web clients were assembled based on two HTTP request fields combined: an IP address and a user agent string; afterwards, they were divided into sessions on the assumption of a minimum 30-minute inter-session time gap [4,15,18,21]. Extremely short sessions, containing less than three HTTP requests, were treated as outliers and removed from the session set.

Based on request fields, each session was described with a number of attributes (features), aiming to reflect differences in traffic patterns of Web bots and legitimate Web users. The initial feature set was developed after extensive analysis of previous bot detection studies, and contained 50 features, summarized in Table 1. It includes key summary session statistics, session characteristics related to the type and volume of resource requested, statistics related to the temporal access, HTTP response statuses, and referrer fields.

For some user sessions, especially the very short ones, some feature values were missing. We eliminated features that were missing for more than 50% of sessions (*iPagAvg*, *iPagSd*, *iEmbSd*). Missing values of the remaining features were substituted with mean feature values computed for all the sessions. Furthermore, features with zeros for all the sessions were removed (*%asc*, *%othFileType*, *trapFile*). In the end, the baseline, non-selective feature set, *F*, contained 44 features.

4.1.1. Ground truth session labeling

Each reconstructed session was assigned a class label (ground truth label): *1*, corresponding to a bot, or *0*, representing a human. For this purpose, two online databases of expert-marked user agents and IP addresses of known bots and Web browsers were used: *Udger* [33] and *User-agents* [34]; moreover, some heuristic rules were also applied. Session labeling rules were as follows [35]:

1. Label *1* was assigned if at least one of the following conditions was met:
   - the client user agent was marked in the *User-agents* database as a robot or in the *Udger* database as "crawler", "e-mail client", "library", "validator", "multimedia player", or "offline browser";
   - the user agent contained a keyword suggesting a bot ("archive", "crawler", "hack", "harvest", "robot", "scrap", "search", "spider", "track", "trap", "worm", etc.);
   - the client IP address was marked in the *Udger* database as "crawler", "fake crawler", "known attack source – http", "known attack source – mail", or "known attack source – ssh";
   - at least one bot indication condition was true: no image requests, no page requests, all requests with empty referrers, all page requests with empty referrers, all requests erroneous (with 4xx code), all requests of type HEAD, *robots.txt* file requested in session.
2. Label *0* was assigned if the user agent was marked in the *Udger* database as "browser" or "mobile browser";
3. Otherwise, the session remained unlabeled.

The usage of the *Udger* database has an additional advantage that the majority of sessions in our data set have not only a class label assigned but also a category, name, and version of the client software. This information may be then used in the in-depth analysis of generated session profiles.

4.1.2. Feature scoring with Fisher score

Solving machine learning problems with high-dimensional data often involves an additional step of feature selection. Identification and elimination of irrelevant features allows one to increase classifier learning capabilities, simplify the model interpretability, and reduce the computational cost of classification. The problem of automatic feature selection may be formulated as follows: find the best possible subset F_r from a large, baseline feature set F, $F_r \subset F$, defining a lower-dimensional feature space in which learning is more reliable with the limited training data set [36].

To assess the robustness of individual session features in discriminating bots from humans, the Fisher Score (FS) [36,37] method is applied. Fisher Score belongs to one of the most widely used techniques for supervised feature selection [37]. As a filter-based method, it ranks features during a pre-processing step, irrespectively of the learning algorithm, after which a subset of features with the top ranks is selected to be used in classification. Fisher scores are determined for respective features separately based on the mean and variance of feature values so as to assign high scores to features which have similar values within the classes and highly varied values between the classes.

Let T be the number of session features, $T = |F|$, C be the number of classes under consideration, and N_c be the number of

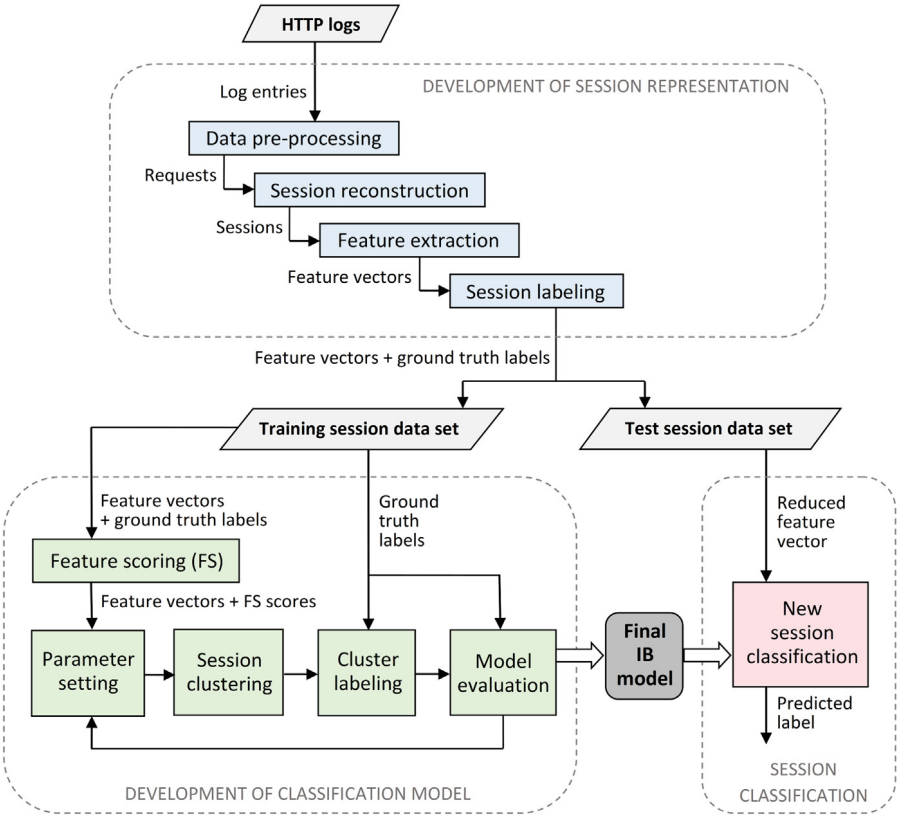

Fig. 2. Flowchart of the proposed approach.

Table 1
Initial set of user session features.

No.	Feature names	Data type	Description
1	*totalRequests*	Int	Total number of requests
2	*totalPages*	Int	Total number of page requests
3	*duration*	Int	Session duration (in seconds)
4	*totalVolume*	Double	Total volume of data (in kilobytes) sent to client
5–13	*%page*, *%image*, *%mult*, *%asc*,*%zip*, *%binDoc*, *%binExe*, *%multAscZipBin*, *%othFileType*	Double	Percentages of requests for specific file types (pages, images, multimedia files, text files, zipped files, binary document and program files, and other file types)
14	*swiFactFileType*	Double	Switching factor on file type
15–16	*embToPageRatio*, *imageToPageRatio*	Double	Ratios of embedded file requests to page requests and image requests to page requests
17	*maxEmbBarrage*	Int	Maximum number of embedded requests per page
18	*%repeatedUri*	Double	Percentage of repeated requests
19–20	*avgRUri*, *maxRUri*	Double	Average and maximum numbers of requests per file
21–22	*robots.txt*, *trapFile*	bool	robots.txt/other hidden file requested in session
23–24	*volAvg*, *volSd*	Double	Average and standard deviation of transfer volume
25–28	*%get*, *%head*, *%post*, *%othMethod*	Double	Percentages of requests made with various HTTP methods: GET, HEAD, POST, and other
29–37	*%200*, *%2xx*, *%2xxWithout200*, *%301*, *%302*, *%304*, *%3xx*, *%4xx*, *%5xx*,	Double	Percentages of requests with various HTTP codes: successful requests (200 and other 2xx codes), redirection (3xx codes), client errors (4xx codes), server errors(5xx codes)
38–39	*%empRefReq*, *%empRefPag*	Double	Percentages of requests and page requests with empty referrers
40	*swiFactEmpRef*	Double	Switching factor on empty referrer
41–43	*msClickRate3*, *msClickRate6*, *msClickRate12*	Int	Maximum numbers of page requests accomplished within 3, 6, and 12 s
44	*%night*	Double	Percentage of requests sent between 12 and 7 a.m.
45–50	*iReqAvg*, *iReqSd*, *iPagAvg*, *iPagSd*, *iEmbAvg*, *iEmbSd*	Double	Averages and standard deviations of request inter-arrival time (in seconds) for: all requests, page requests, and embedded file requests

samples of class c, $c \in \{0, 1\}$. A Fisher score for the jth feature, $j = 1, 2, \ldots, T$, is computed according to the following formula:

$$FS_j = \frac{\sum_{c=1}^{C} N_c \left(\mu_c^j - \mu^j\right)^2}{\sum_{c=1}^{C} N_c (\sigma_c^j)^2}, \tag{1}$$

where μ_c^j and σ_c^j are the mean and standard deviation of the jth feature for the cth class, respectively, and μ^j is the mean of the jth feature for all the samples. All the features are ranked in a descending order according to the determined scores, giving an ordered feature set $F' = (o_1, \ldots, o_T)$, where o_1 is the index of the feature with the best score.

Given the feature ranking F', a specific subset of r top ranked features may be then selected. We determine the value of r experimentally by assessing its impact on the clustering performance (see Section 5.3.1).

4.2. Experimental setup for clustering and classification

An experimental verification of the proposed approach was performed by allocating data into mutually disjoint sets: a *training set*, containing 500 sessions in total, and a *test set*, containing 1000 sessions, each set with the equal numbers of randomly drawn bot and human sessions. A training data set was used in session clustering as a base to develop and learn a classification model. A test set was used to verify the classification model performance on previously unseen data. The decision to set the training set size to 500 was reached as a result of various experiments aiming to identify the smallest possible number of sessions which leads to good clustering results, starting from 1000 sessions (see Section 5.3.3). Regarding the test set size, the initial cardinality of 1000 was not reduced as the testing procedure is much simpler and not so computationally expensive as the model training.

Each experiment was carried out 100 times for different independent random draws from the whole session data set X_{all} (100 different, mutually disjoint training and test sets). Resulting performance rates were averaged and reported in the paper.

Performance of aIB was additionally compared with performance of two other agglomerative clustering algorithms: complete-linkage and Ward.D2, as well as with two common benchmarking clustering techniques: NMF (Non-negative Matrix Factorization) and k-means with Euclidean distance measure (see Section 5.5.3). The experimental setup for the reference algorithms was the same as for aIB: experiments were also run 100 times, for exactly the same training and test session sets. Additionally, since k-means and NMF are known to be sensitive to random data points picked as initial centroids (cluster centers), for each number of clusters the algorithms were additionally repeated 100 times with different random initial picks; finally, results were averaged.

4.3. Development of a classification model based on unsupervised learning

To discover hidden patterns intrinsic for bot and human sessions of different types, we apply the Information Bottleneck (IB) method. More precisely, the agglomerative Information Bottleneck (aIB) algorithm is used to cluster user sessions represented as reduced feature vectors.

4.3.1. The agglomerative information bottleneck

The Information Bottleneck method was proposed in [38], addressing the problem of finding such a compact representation of some random variable X that is relevant for prediction of another variable Y. This task can be reformulated as the problem of finding a compact representation $\tilde{X}$ of random variable X, minimizing the mutual information $I(X;\tilde{X})$ between X and $\tilde{X}$ under a constraint that the mutual information $I(\tilde{X};Y)$ between $\tilde{X}$ and another random variable Y is above a certain level. $\tilde{X}$ describes the clustering and Y, called a relevance variable, describes target features for clustering X.

The mutual information between the random variables X and $\tilde{X}$ is defined as the symmetric functional of their joint distribution:

$$I(X;\tilde{X}) = \sum_{x\in X,\tilde{x}\in\tilde{X}} p(x,\tilde{x}) \log \frac{p(x,\tilde{x})}{p(x)p(\tilde{x})}, \tag{2}$$

where $p(x,\tilde{x})$ is the joint probability distribution for X and $\tilde{X}$, $p(x)$ and $p(\tilde{x})$ are the marginal probability distributions for X and $\tilde{X}$, respectively.

This is a constrained optimization problem which cannot be solved analytically; instead, it has yielded a number of iterative algorithms that converge to a compact representation $\tilde{X}$.

We apply the agglomerative Information Bottleneck algorithm [39], which generates a hierarchical bottom-up clustering tree. The algorithm starts with the initial partition where $\tilde{X} = X$, i.e., each value of X is assigned to a unique cluster in $\tilde{X}$. Then, the number of clusters is reduced iteratively by merging two selected clusters at each step. For the aIB algorithm, the formulation of the constrained optimization problem can be expressed as a problem of maximization of the IB functional [40]:

$$\mathcal{L}_{max} = I(\tilde{x};y) - \beta^{-1} I(\tilde{x};x), \tag{3}$$

where β is a Lagrange multiplier.

In our scenario, a random variable X corresponds to user sessions and a random variable Y corresponds to session features. Our goal is to construct the assignment of user sessions into a set of clusters $\tilde{X}$ that preserves as much mutual information between the sessions and the session features as possible. The joint probability distribution matrix $p(x,y)$ is calculated from the feature vectors representing user sessions. Thus, despite normalizing session feature values into the range (0, 1), all the values in the matrix have to be additionally normalized to be mapped into joint probabilities (the sum of all normalized feature values for all the sessions is 1). This normalization is performed on the assumption that the marginal probability distribution of random variable Y is uniform and that the original feature value is proportional to the corresponding probability assigned.

The key point of the aIB algorithm is selection of clusters to be merged at each subsequent step. Since the goal is to maximize (3), one needs to find two clusters $\tilde{x}_i$ and $\tilde{x}_j$ to be merged into a new cluster x^*, which minimizes the cost given by:

$$\Delta\mathcal{L}_{max}(\tilde{x}_i,\tilde{x}_j) = \mathcal{L}_{max}^{(before)} - \mathcal{L}_{max}^{(after)}, \tag{4}$$

where $\mathcal{L}_{max}^{(before)}$ and $\mathcal{L}_{max}^{(after)}$ are the IB functionals before and after merging the clusters, respectively. The greedy procedure checks all potential pairs of clusters for merging and selects the best one. The value of $\Delta\mathcal{L}_{max}(\tilde{x}_i,\tilde{x}_j)$ is calculated by the following formula [39]:

$$\Delta\mathcal{L}_{max}(\tilde{x}_i,\tilde{x}_j) = [p(\tilde{x}_i) + p(\tilde{x}_j)] * d(\tilde{x}_i,\tilde{x}_j), \tag{5}$$

where:

$$d(\tilde{x}_i,\tilde{x}_j) = JS_\pi[p(y|\tilde{x}_i), p(y|\tilde{x}_j)] - \beta^{-1} JS_\pi[p(x|\tilde{x}_i), p(x|\tilde{x}_j)], \tag{6}$$

where $JS_\pi[p_i,p_j]$ is called the Jensen–Shannon (JS) divergence, given by:

$$JS_\pi[p_i,p_j] = H[p^*] - \pi_i H[p_i] - \pi_j H[p_j], \tag{7}$$

where

$$p^* = \pi_i p_i + \pi_j p_j, \tag{8}$$

$$\pi = \{\pi_i,\pi_j\} = \left\{\frac{p(\tilde{x}_i)}{p(\tilde{x}^*)}, \frac{p(\tilde{x}_j)}{p(\tilde{x}^*)}\right\}. \tag{9}$$

The procedure follows until the assumed number of clusters is achieved or $\tilde{X}$ degenerates into a single cluster. The result of the algorithm is a tree of clusters describing a range of clustering solutions at all possible resolutions.

The aIB algorithm provides hard clustering, in which each session is assigned to exactly one cluster. It means that conditional

probabilities $p(\tilde{x}|x)$ can be only equal to 1 or 0 and $I(X;\tilde{X}) = H(\tilde{X})$. Thus, (3) may be rewritten as:

$$\mathcal{L}_{max} = I(\tilde{x}; y) - \beta^{-1} H(\tilde{x}). \tag{10}$$

Therefore, we aim at minimizing $H(\tilde{X})$ while maintaining $I(\tilde{X}, Y)$ as high as possible. The algorithm is illustrated in Fig. 3.

$H(\tilde{X})$ decreases when the probability distribution becomes more differentiated, which results in bigger differences in cluster sizes. Thus, increasing the value of β^{-1} will lead to more differentiated cluster sizes, typically one big cluster and many smaller clusters. Since we try to maintain the highest possible value of $I(\tilde{X}, Y)$, such a big cluster will contain the values of X which are less informative about Y. By increasing β^{-1} we can apply a kind of a filter to remove from smaller clusters values of X less informative about Y. We experiment with various values of β^{-1} to investigate an impact of this parameter on results of user session clustering (see Section 5.3.2).

For a detailed discussion on the IB theory and algorithms we refer to [40].

4.3.2. Clustering performance measure

We investigate different target numbers of clusters (k) to find a reasonable trade-off between the clustering quality and model interpretability. To assess clustering performance for a given k, generated clusters were labeled with majority class ground truth labels. This allowed us to evaluate the clustering efficiency in terms of separation of bots from humans by computing a degree of correspondence between the clusters' labels and the sessions' ground truth labels. A degree to which each cluster consists of objects of a single class is expressed with the entropy measure [41]. The entropy of cluster i, $i = 1, 2, .., k$, is given by the formula:

$$H_i = -\sum_{c=1}^{2} p_{i,c} \ \log_2 p_{i,c}, \tag{11}$$

where $p_{i,c}$ is the probability that a session in cluster i belongs to class c, $p_{i,c} = l_{i,c}/l_i$, where $l_{i,c}$ is the number of sessions of class c in cluster i and l_i is the number of all sessions in cluster i.

The total clustering entropy for all k clusters is given by:

$$H = \sum_{i=1}^{k} \frac{l_i}{l} H_i, \tag{12}$$

where l_i is the number of sessions in cluster i, l is the total number of sessions, and H_i is the entropy of cluster i.

We ensure that in each of 100 experiments the numbers of bot and human sessions in a training data set are equal. Since a random variable (class label) is binary, interpretation of the entropy measure is straightforward: entropy is in the range [0, 1], where 0 means the ideal separation of robots and humans amongst all the clusters whereas 1 means that each cluster contains 50% of instances of each class.

4.4. Classification of new sessions based on unsupervised learning model

Generated clusters reveal the initially hidden knowledge about individual session profiles. A result of session clustering for a given k is a session model, which can be further used to categorize new sessions.

4.4.1. Cluster labeling and session classification

To be able to use the generated clusters in classification, it is necessary to label the clusters based on ground truth labels of sessions included in them. We propose two ways of cluster labeling for session classification:

1. *Majority class labeling*
   A cluster label is equal to the majority class label, i.e. the one most represented among sessions in the cluster. Thus, there are two possible cluster labels: *1* (bot) and *0* (human).
2. *Threshold-based labeling*
   Since besides pure bot and human clusters there are also the mixed ones, containing sessions of both classes assorted in different proportions, we propose to introduce an additional cluster label, $mixed_X$, for the mixed clusters containing a share of majority class sessions below some threshold, $T_{mix} = X$. Given $T_{mix} = X$, a cluster containing more than X% of sessions of class c, $c \in \{0, 1\}$, is assigned a class label c. Otherwise, a cluster is labeled as $mixed_X$. For example, for $T_{mix} = 90$ there will be the third cluster label, $mixed_{90}$, assigned to all mixed clusters with less than 90% of majority class sessions (e.g., to a cluster containing 70% of bots and 30% of humans).
   The additional label allows us to flag the most ambiguous mixed clusters, which can be then analyzed in more detail, e.g., in terms of the possibility of containing camouflaged bot sessions, which had been assigned improper ground truth labels. Furthermore, the third label will cause an increase in classification accuracy for sessions assigned to pure clusters.

Let x^* be a new session, represented with a reduced feature vector. In the case of the aIB-based model, classification of x^* may be done by comparing its feature values normalized to probabilities with all k clusters and selection of the cluster for which $\Delta\mathcal{L}_{max}(x^*, \tilde{x}_i)$ is minimized, $i = 1, 2, \ldots, k$ (like when selecting a cluster for merging by the aIB algorithm).

In the case of the k-means-based model, the target cluster for x^* is selected as the one with the minimum Euclidean distance from the x^* reduced feature vector to the cluster centroid.

4.4.2. Classification performance measures

Each classification result for a single session may be *positive* (when a predicted label is *1*) or *negative* (a predicted label is *0*). Furthermore, by comparing the session ground truth label and the predicted label, a classification outcome may be *true* (when both labels are the same) or *false* (the labels are different).

Let *TP* be the number of true positives (correctly recognized bots), *TN* – the number of true negatives (correctly identified humans), *FP* – the number of false positives (humans mistakenly classified bots), and *FN* – the number of false negatives (bots mistakenly taken for humans). Total classification results may be expressed with the standard performance measures of recall, precision, accuracy, and F1.

Recall (sensitivity) is a fraction of correctly recognized bots among all bots:

$$Recall = \frac{TP}{TP + FN}. \tag{13}$$

Precision is a fraction of correctly recognized bots among all positive classifications:

$$Precision = \frac{TP}{TP + FP}. \tag{14}$$

Accuracy reflects the global classification quality as the fraction of all correct classifications:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}. \tag{15}$$

F1 (F-score or F-measure) also represents the overall classifier performance, expressed as the harmonic mean of recall and precision:

$$F1 = \frac{2 \cdot Recall \cdot Precision}{Recall + Precision}. \tag{16}$$

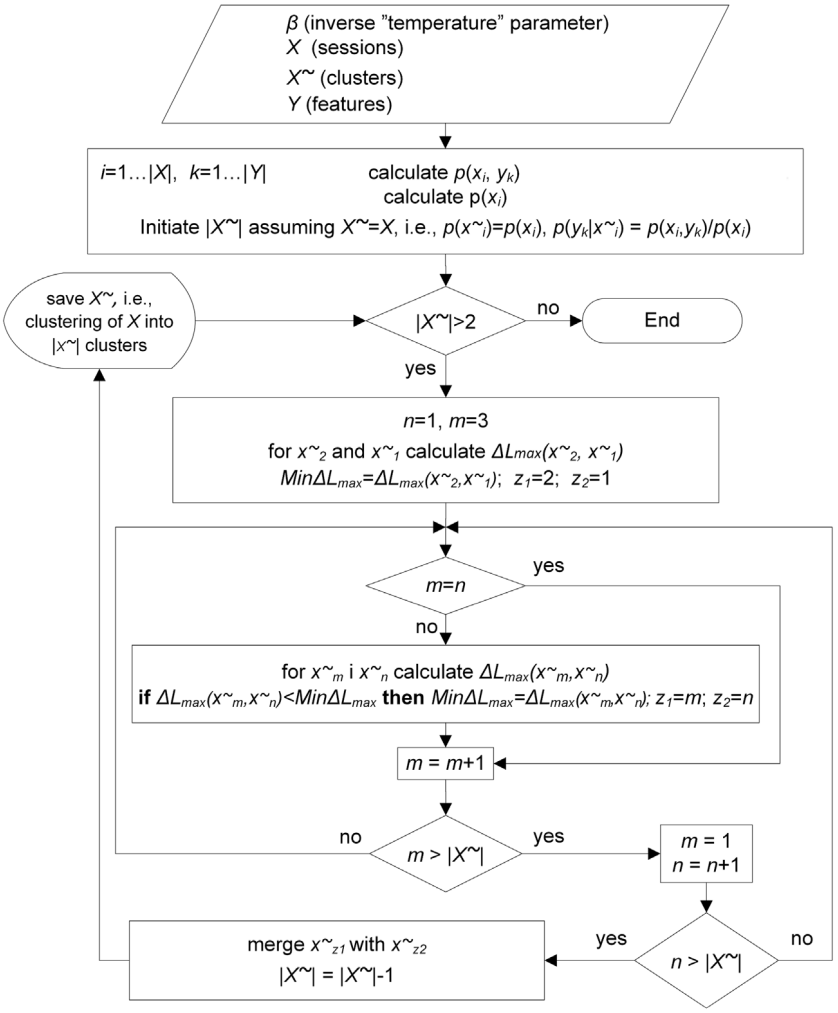

Fig. 3. Agglomerative Information Bottleneck algorithm.

5. Results and discussion

5.1. Data set description

Data used in the analysis was server access data logged for a tourist Web store (the retailer identity cannot be revealed in the paper due to a non-disclosure agreement). Logs spanned a period from 12 to 25 May, 2015 and contained 387 511 requests, corresponding to 6762 user sessions. After excluding extremely short sessions and assigning class labels to the remaining ones, the input data set contained 4922 sessions: 2443 bots and 2479 humans.

5.2. Feature scoring results

All the 44 session features were assigned Fisher scores and ranked in a descending order according to them. Fig. 4 shows the resulting ranking for the features scored higher than 0.02. It can be noticed that there are two key features highly discriminating sessions of both classes: the shares of requests and page requests with empty referrers; they are characterized by Fisher scores of 3.2 and 1.7, respectively. The subsequent five features, with the scores in the range of 0.58–0.69, are also important: the percentages of page requests and image requests, embedded-to-page ratio, image-to-page ratio, and the maximum number of embedded requests per page.

It is worth pointing out that an access to *robots.txt* file in session, which is commonly supposed to be an indication of a bot visit, is ranked only at the 8th position, with a Fisher score of 0.31. Moreover, such features considered common for bots as the shares of requests made during the night hours, requests of type HEAD, and requests with status 4xx turned out to be insignificant for the analyzed e-commerce website.

5.3. Model tuning results

5.3.1. Tuning the number of session features

Our approach to determine dimensionality of the best feature subset, F_r, relies on the experimental evaluation of the clustering quality (mean entropy value) for different numbers of session features from the ordered set F'. Starting from $r = 1$ (i.e., taking only one, the highest-ranked session feature) to T (taking all the features), each ordered subset $(o_1, \ldots, o_r)$ of F' was evaluated in session clustering. For each number of features r, a series of experiments was performed for different k.

Clustering results as a function of the number of features, averaged over 100 experiments run on different data sets (500 session-subsets of X_{all}), are shown in Fig. 5. It can be seen that in general, the higher number of clusters is given, the better results are generated (i.e., the lower entropy rates are achieved). Despite very small values of $k < 4$, for a given number of clusters entropy reaches a minimum for about six features. This observation suggests a procedure for selecting the most suitable,

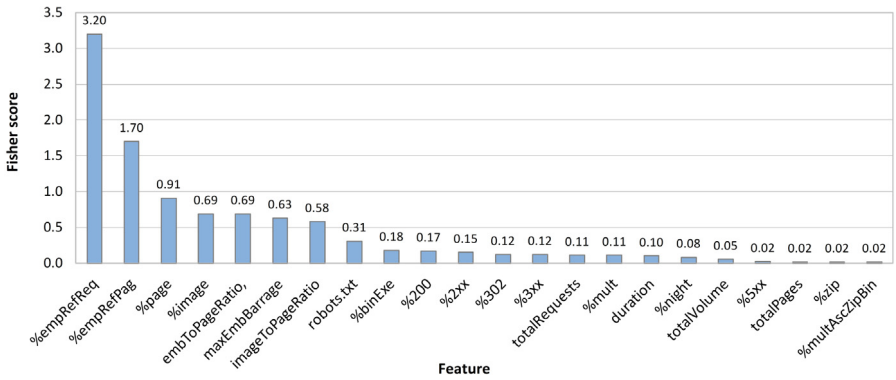

Fig. 4. Fisher score ranking (session features with scores higher than 0.02 are shown).

compact set of features for a given k: first, order all the features according to decreasing values of their Fisher scores and then, take the number of features that minimizes the entropy value.

Another interesting outcome from Fig. 5 is that for $k > 4$ taking many irrelevant features into account significantly deteriorates the clustering quality: e.g., for $k = 20$, the entropy increases from 0.08 for six features to 0.28 for 24 features.

As a result of the aforementioned analysis, the number of features describing user sessions was reduced from 44 to six ($|F_r| = 6$). This value has been used in experiments discussed in the paper.

5.3.2. Tuning the aIB algorithm

As it was noted in Section 4.3.1, results of aIB depend on parameter β. For the hard clustering version of the algorithm, applied in this study, this dependence is expressed by (10).

An impact of β on clustering results was investigated experimentally. Fig. 6 plots entropy rates achieved for different β as a function of the number of clusters. One can notice the unsuitability of very large values of β^{-1}, which result in high entropy for the whole range of k. Going to lower levels of β^{-1}, one may observe the following tendency: higher values of β^{-1} lead to better entropy rates for lower k and lower values — for higher k. After the profound investigation we found that when the number of clusters does not exceed 20, $\beta^{-1} = 0.1$ is preferred but for higher k $\beta^{-1} = 0.01$ gives better results. Therefore, it is recommended to adjust the value of parameter β for a scenario under consideration depending on a target number of clusters.

5.3.3. Tuning a training set size

Using a compact training set for session clustering has many advantages, including lower computational cost, i.e., shorter computing time. Obviously, a training set containing too few samples may not encompass all types of user behaviors and thus, may not be representative of the whole session set. We determine this parameter value experimentally by investigating the impact of the training set size on the clustering quality. The results summarize 100 experiment runs for different random subsets of X_{all}, the numbers of bot and human sessions in each training set are equal.

Fig. 7 plots entropy as a function of a training set size for six and twelve clusters. Entropy rates are extremely low for very small numbers of samples — it has to be noted, however, that this is due to the agglomerative nature of the aIB algorithm; e.g., when we have only six clusters, the entropy for six sample sessions is equal to 0 because each session is assigned to a separate cluster. Thus, conclusions should be drawn based on the observation of larger numbers of samples. One can notice that starting from about 300 samples in a training set, the average and the standard deviation of entropy stabilize so values above 300 samples seem to be adequate for the training set size according to this criterion. We have selected the training set size of 500 sessions, providing a safety margin of 200 samples.

5.4. Clustering results

Session clustering was carried out for a wide range of k to estimate the minimum number of clusters for which the clustering quality stabilizes. Fig. 8 presents entropy rates for k ranging from 1 to 50 ($\beta^{-1} = 0.1$). Entropy is plotted not only for 6-feature session vectors, as determined in Section 5.3.1, but also for $|F_r|$ equal to 4, 8, and 10. Results confirm our choice of the most appropriate number of session features as six: for this case the entropy rates are the lowest almost throughout the whole range of k. For a very coarse clustering, the efficiency of aIB in separating bots from humans is low but it significantly increases with the number of clusters; for $|F_r| = 6$ it stabilizes for about 18 clusters at the level of about 0.1.

By applying unsupervised learning to user sessions, we expect to identify more Web traffic profiles than simply robots and humans. In this subsection we analyze the clustering efficiency for a relatively small number of clusters that preserves a good split between bot and human sessions; we keep the number of discovered hidden behavioral patterns reasonable but as small as possible for an effective in-depth analysis by a human expert.

The left side of Fig. 9 illustrates the evolution of dataset partitioning from 2 to 6 clusters — visualization of the cluster content reflects a user session split between robots and humans. The right side of the figure shows the corresponding probability distributions for session features in the clusters — they may be treated as reference feature values for individual clusters, which allows us to characterize the discovered session profiles with respect to these features.

We present the results for one of the best experimental cases (with the lowest entropy scores) from among all 100 experiment runs to show a general trend of improving efficiency of unsupervised learning in discovering user profiles with the increase in the number of clusters.

It can be seen that even for only two clusters (Fig. 9, $k = 2$, left) there is a clear split of sessions between the two classes, with one pure cluster, composed of robot sessions only (cluster no. 2), and the second one, which is mixed but clearly dominated by human sessions (cluster no. 1). Visualization of the corresponding feature probabilities (Fig. 9, $k = 2$, right) confirms the well-known discriminants of humans and bots. In human sessions there are much more images than page files requested (probability for *%page* is much lower than for *%image*), page files are accompanied by many embedded objects, not only images (high probability for *embToPageRatio*), and the maximum number of embedded objects per page is very large (high probability for *maxEmbBarrage*). On

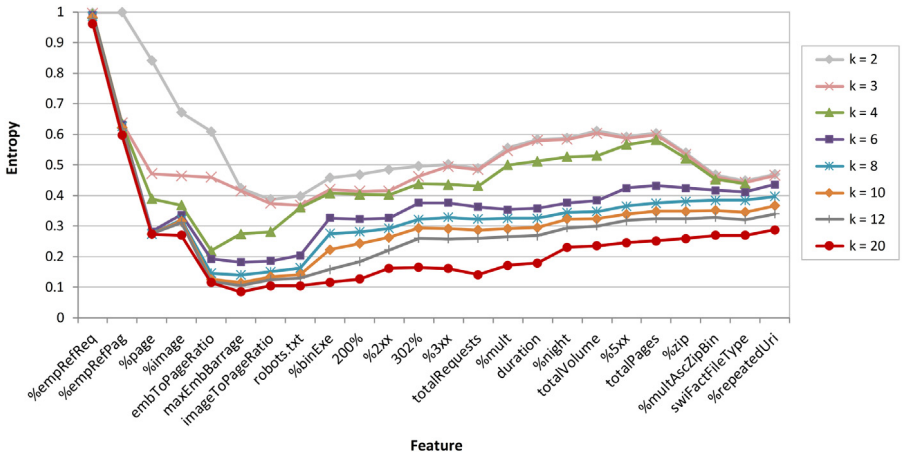

Fig. 5. Clustering entropy as a function of the number of session features taken into account (cumulatively) for different numbers of clusters.

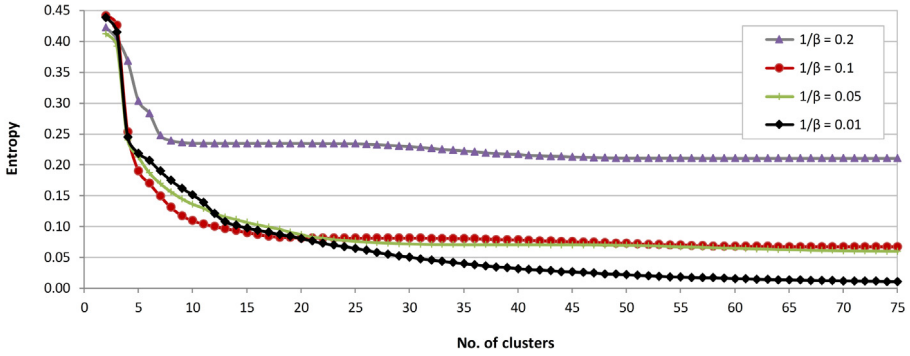

Fig. 6. Impact of the aIB parameter β on the clustering entropy.

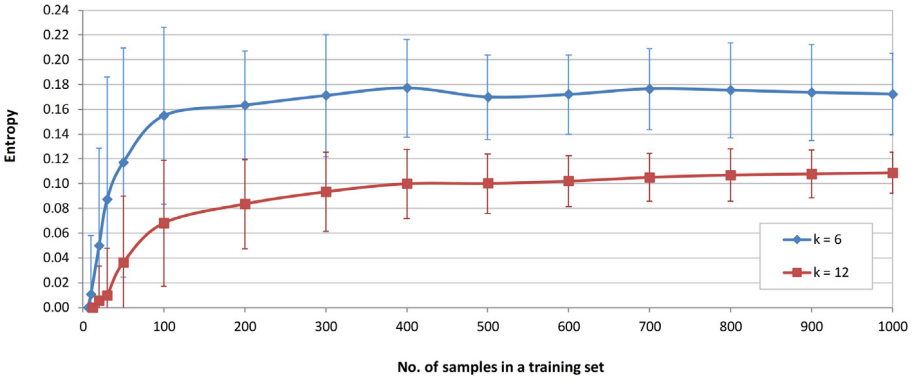

Fig. 7. Impact of the training set size on the clustering entropy.

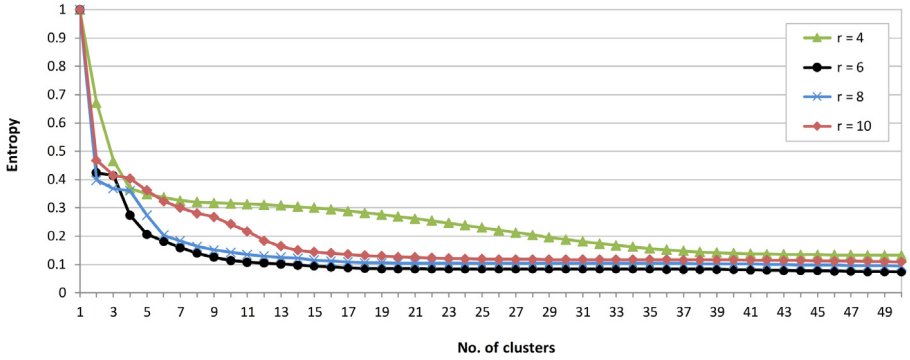

Fig. 8. Clustering entropy as a function of the number of clusters.

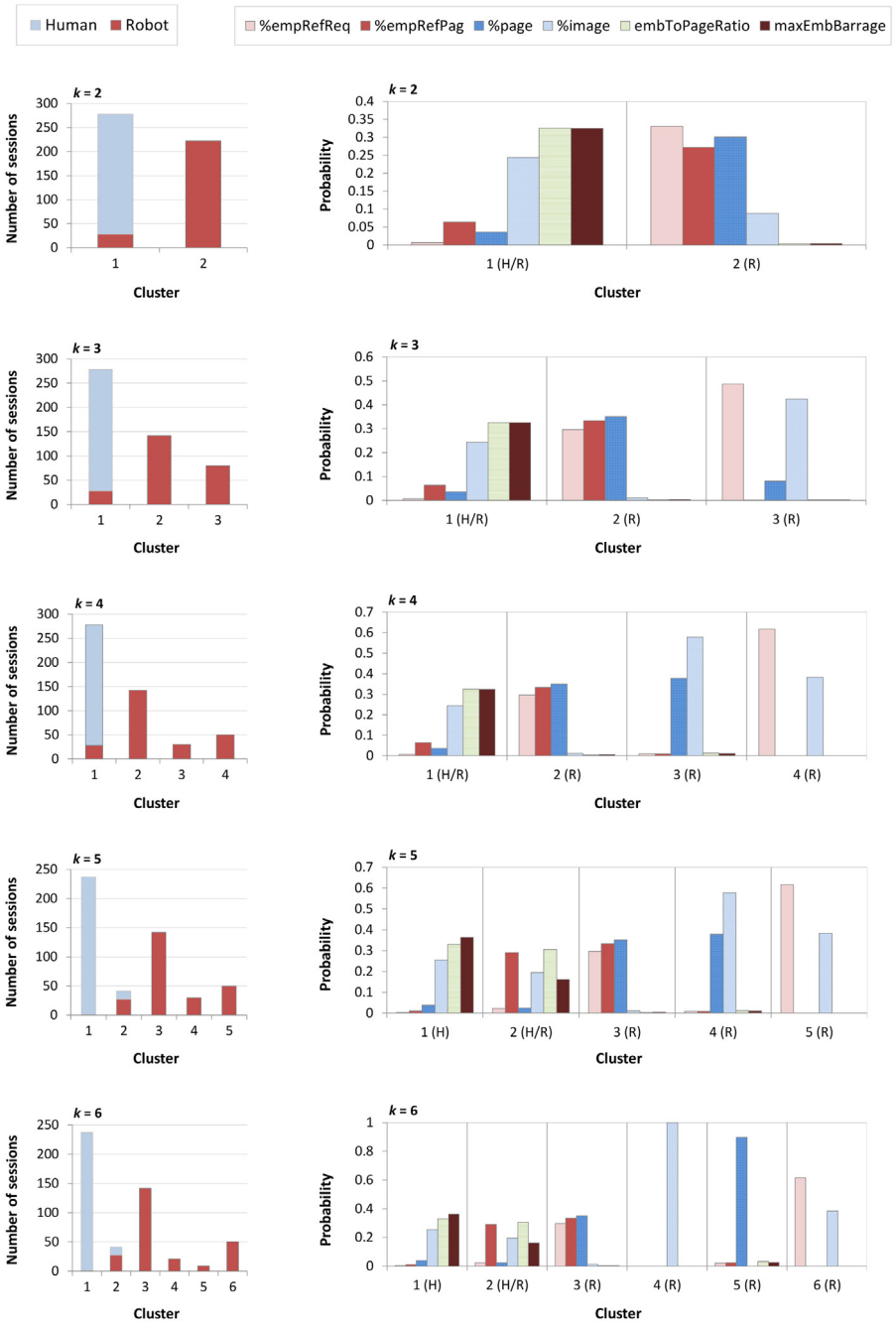

Fig. 9. Example distribution of bot and human sessions among clusters (left) and probability distributions of session features in the clusters (right); $k = 2, \ldots, 6$.

the other hand, robot sessions are characterized by many HTTP requests and page requests with empty referrers (*%empRefReq* and *%empRefPag*) and page files are requested significantly more often than images; at the same time, probabilities for *embToPageRatio* and *maxEmbBarrage* are close to zero. Since this session partition is very coarse and there is still a great session diversity inside both clusters, probabilities for the individual features are rather low – they do not exceed 0.34. The clustering entropy is equal to 0.262.

We can observe that after increasing the number of clusters to three and four a pure bot segment (cluster no. 2 in Fig. 9, $k = 2$, left) is further divided into several bot subgroups while the largest, human-dominated cluster remains unchanged. For $k = 5$ the largest cluster is just divided and the overwhelming majority of human sessions are gathered in one segment (cluster no. 1 in Fig. 9, $k = 5$, left).

Clustering for $k = 6$ leads to generating one large human cluster, four pure robot clusters, and one mixed, robot-biased cluster; this partition is characterized by entropy of 0.095. The discovered session profiles are more specific than for lower k, with probabilities for some features approaching 1. In Fig. 9, $k = 6$, right we can observe that characteristics of the human cluster (cluster no. 1) changed only slightly compared to cluster no. 1 for $k = 2$: the main difference lies in much lower probabilities for *%empRefReq* and *%empRefPag* due to displacement of robot sessions to cluster no. 2 (the mixed one). This mixed cluster is very similar to the human one in terms of the features *%page*, *%image*, and *embToPageRatio* (slightly lower probabilities); the main difference is a lower value for *maxEmbBarrage* and a significantly

higher value for *%empRefPag* (percentage of page requests with empty referrers). These characteristics suggest that cluster no. 2 may contain some masquerading bots, which efficiently imitate humans.

Two pure bot clusters (no. 3 and 5) seem to contain crawlers indexing the textual Web content. Cluster no. 5 groups sessions in which a great majority of requests were for page files (probability for *%page* is 0.9) while images were not requested at all (although there were very few accesses to embedded files of other types, which is indicated by non-zero probabilities for *embToPageRatio* and *maxEmbBarrage*). Cluster no. 3 assembles sessions with the majority of accesses to page files and very few requests for images and other embedded objects as well; a clear discriminants of these sessions are almost all requests with empty referrers (probabilities for *%empRefReq* and *%empRefPag*, and *%page* are at the similar level). Two other clear bot clusters (no. 4 and 6) are evidently related to image crawlers, targeted on searching for graphics files only (no other file types were accessed in these sessions). Sessions in cluster no. 6 have additionally a large share of requests with empty referrers.

Our interpretation of the clustering results may be confronted with additional information on the Web clients, i.e., categories, names, and versions of the software used to accomplish the sessions. The cluster content for $k = 6$ broken down by client categories is presented in Table 2 and by client names – in Table 3. It should be noted that not for all sessions the categories and names were provided.

It is evident that all the legitimate user sessions (client categories "Browser" and "Mobile browser") were assigned to the human (236 sessions) or mixed (14 sessions) clusters; there is no clear difference in classification results for conventional browsers and mobile browsers. However, for $k = 6$ these two clusters contain also some bot sessions, including a large part (24 samples) of sessions having all page requests with empty referrers (although 13 sessions of this type were assigned to the text crawler cluster, no. 3).

It can be seen that a great majority of search engine bots, acting on behalf of Bing, Exalead, Google, Qwant, Sogou, and Yandex search engines, were assigned to two pure bot clusters: the text crawler cluster no. 3 (84 sessions) and the image crawler cluster no. 6 (27 sessions); a few Google tools, however, were allocated to some other clusters. So most of known browsers' agents were assigned to cluster no. 3 and 6 whereas most of unknown bots dropped in cluster no. 4 and 5. These findings show that aIB clustering without any knowledge of session labels allowed us to filter out known search engine bots from unidentified image or text indexing bots based solely on some traffic patterns. Furthermore, the selected session features allowed gathering all marketing agents, working for AddThis, Ahrefs, MixRank, Majestic, and SMT, in the text crawler cluster no. 3 and all e-mail clients (Gmail and Windows Live Mail) in the image crawler cluster no. 6.

To sum up the analysis of the clustering results, we can conclude that there is a pretty clear separation of human and bot sessions of different types even for only six subgroups; however, there is still a significant part of bot sessions with similar traffic patterns to the human ones. Increasing the number of clusters above six leads to further specialization of generated profiles and lower entropy rates (as was shown in Fig. 8) but the in-depth analysis and interpretation of the profiles becomes more difficult. Our general observation of this evolution for bigger k is that large clusters remain large whereas a number of mini-clusters containing only a few sessions arise.

5.5. Classification results

5.5.1. Binary classifier

To assess the usefulness of the model developed via unsupervised learning and verify classifier learning capabilities, the model was applied to categorize new user sessions with respect to the binary criterion (bot or human).

Let us consider a simple example of assigning sessions into one of two clusters (the case for $k = 2$ in Fig. 9). Cluster no. 1, which is a mixed cluster (H/R), is characterized by the following values of a feature probability vector: [0.0065, 0.0639, 0.0356, 0.2435, 0.3256, 0.3249] whereas a robot cluster no. 2 – by a vector: [0.3312, 0.2722, 0.3014, 0.0877, 0.0034, 0.0041]. In such a case, a session driven by a Chrome Web browser, represented by a feature vector [0, 0, 3.13, 84.38, 31, 31], will be assigned to cluster 1 and marked as a human; on the other hand, a Googlebot session, represented by a feature vector [100, 100, 86.67, 0, 0.15, 2] will be classified to cluster 2 and marked as a bot.

We start with investigating classification accuracy depending on the number of clusters and the number of samples in the training session data set. For plot clarity, results are visualized only for the aIB-based model with majority class cluster labeling for $\beta^{-1} = 0.1$.

Fig. 10 presents accuracy for different training set sizes as a function of k. It can be seen that for various sizes the accuracy scores stabilize at about 15 clusters and are the highest for 500 session samples. Fig. 11 shows accuracy for 6, 10, and 20 clusters as a function of the training set size. It reaffirms that accuracy achieved for about 500 samples does not increase with enlargement of the training set size regardless of k (accuracy rates for the set of 1000 sessions are a bit lower than these achieved for 500 sessions). These outcomes confirm our findings from Section 5.3.3 that 500 is the adequate training set size.

Selection of the most appropriate number of clusters based on accuracy is not so obvious. In general, accuracy is high – even for $k = 2$ it exceeds 0.84; it is lower for lower k but above some level of k it decreases slightly again. Fig. 12 shows accuracy rates for classification based on aIB with $\beta^{-1} = 0.01$ (i.e., the version of the algorithm which gives better clustering entropy for higher k), for the number of clusters reaching 500. One can observe that accuracy increases rapidly with k till about 62 clusters, reaches the value of 0.986 for 85 clusters, and then stabilizes almost at the same level. Thus, increasing the number of clusters above 85 will not improve the results but will only increase the computational time needed for session classification.

Fig. 13 presents all four kinds of performance scores for classification based on aIB clustering with $\beta^{-1} = 0.1$ and 0.01 with majority class cluster labeling. Since in all experiments session subsets of both classes were balanced, accuracy and F1 are at very similar levels. Plots of accuracy, F1, and recall partially confirm our findings from Section 5.3.2 that $\beta^{-1} = 0.1$ gives better results for lower k whereas $\beta^{-1} = 0.01$ is better for higher k. A plot of precision, however, indicates the advantage of $\beta^{-1} = 0.1$ regardless of k.

For small numbers of clusters the performance scores are unstable – they stabilize only for k approaching 15. For higher k performance is very high: the maximum values of mean accuracy, achieved for $k = 84$, is 0.986 (1 for the best case experiment run), maximum mean recall (for $k = 49$) is 0.990 (1 for the best case), maximum mean precision (for $k = 139$) is 0.988 (1 for the best case), and maximum mean F1 (for $k = 84$) is 0.986 (0.997 for the best case).

The main weakness of the binary classifier is that it gives much worse results for small numbers of clusters, i.e., for the cases when the cluster content may be easily analyzed by an expert in terms of various user profiles. This drawback may be attributed

Table 2
Cluster content broken down by client categories ($k = 6$).

Client category	Class	Cluster					
		1 (*H*) *Web browser*	2 (*H/R*) *mixed*	3 (*R*) *text crawler*	4 (*R*) *image crawler*	5 (*R*) *text crawler*	6 (*R*) *image crawler*
Browser	*0*	170	9	0	0	0	0
Mobile browser	*0*	66	5	0	0	0	0
Search engine bot	*1*	0	1	83	0	0	26
Screenshot creator	*1*	1	0	0	0	0	0
Tool	*1*	0	2	0	1	0	0
Marketing	*1*	0	0	40	0	0	1
E-mail client	*1*	0	0	0	0	0	9
Library	*1*	0	0	1	0	0	0
Speed tester	*1*	0	0	1	0	0	0
BOT (known, uncategorized)	*1*	0	0	0	0	1	8
BOT (0 pages)	*1*	0	0	0	20	0	6
BOT (0 images)	*1*	0	0	4	0	7	0
BOT (all responses 4xx)	*1*	0	0	0	0	1	0
BOT (all pages without ref.)	*1*	0	24	13	0	0	0

Table 3
Cluster content broken down by client names ($k = 6$).

Client name	Class	Cluster					
		1 (*H*) *Web browser*	2 (*H/R*) *mixed*	3 (*R*) *text crawler*	4 (*R*) *image crawler*	5 (*R*) *text crawler*	6 (*R*) *image crawler*
Chrome	*0*	110	7	0	0	0	0
CoolNovo	*0*	1	0	0	0	0	0
Firefox	*0*	26	0	0	0	0	0
IE	*0*	19	1	0	0	0	0
Opera	*0*	2	1	0	0	0	0
Safari	*0*	12	0	0	0	0	0
Android browser	*0*	11	1	0	0	0	0
Android webview	*0*	1	0	0	0	0	0
Chrome Mobile	*0*	25	1	0	0	0	0
Google App	*0*	3	0	0	0	0	0
IE Mobile	*0*	2	0	0	0	0	0
MIUI Browser	*0*	1	0	0	0	0	0
Mobile Samsung Browser	*0*	1	0	0	0	0	0
Opera Mobile	*0*	0	1	0	0	0	0
Safari mobile	*0*	22	2	0	0	0	0
Bingbot	*1*	0	0	21	0	0	2
Exabot	*1*	0	0	1	0	0	0
Googlebot	*1*	1	3	58	1	0	25
Qwantify	*1*	0	0	1	0	0	0
Sogou spider	*1*	0	0	1	0	0	0
YandexBot	*1*	0	0	2	0	0	0
AddThis.com	*1*	0	0	20	0	0	0
AhrefsBot	*1*	0	0	8	0	0	0
MixrankBot	*1*	0	0	1	0	0	0
MJ12bot	*1*	0	0	10	0	0	0
SMTBot	*1*	0	0	1	0	0	0
Gmail	*1*	0	0	0	0	0	8
Windows Live Mail	*1*	0	0	0	0	0	1
Java	*1*	0	0	1	0	0	0
CFNetwork	*1*	0	0	0	0	0	6
MSIECrawler	*1*	0	0	0	0	1	0
(not given)	*1*	0	24	17	20	8	8

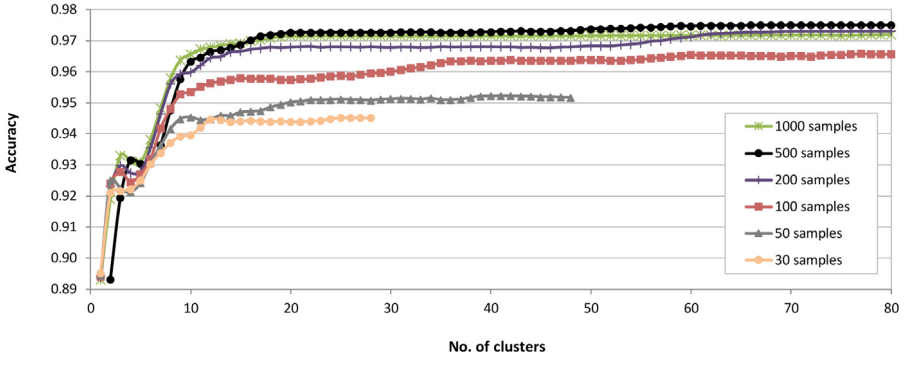

Fig. 10. Classification accuracy for different training set sizes as a function of the number of clusters (aIB, majority class cluster labeling).

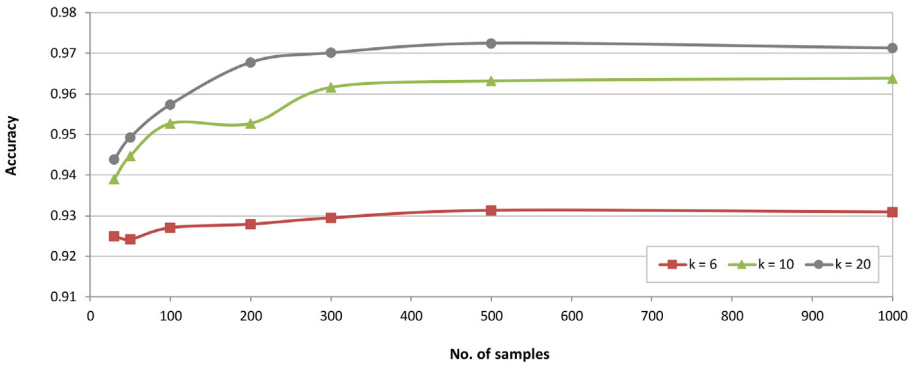

Fig. 11. Classification accuracy for different numbers of clusters as a function of the training set size (aIB, majority class cluster labeling).

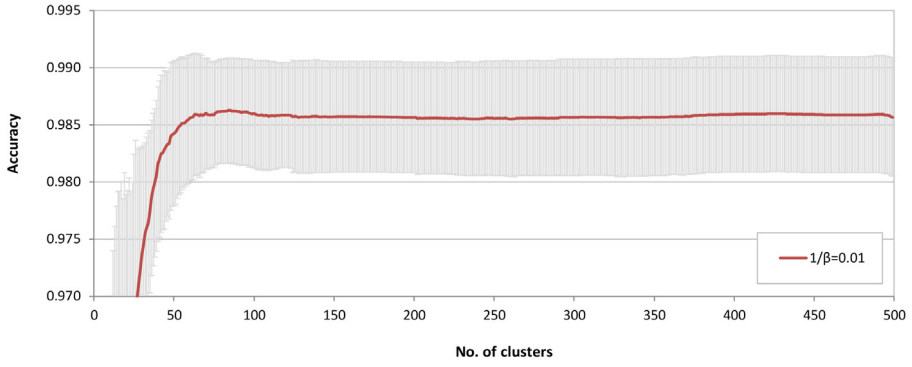

Fig. 12. Accuracy as a function of the number of clusters for k up to 500 (aIB, majority class cluster labeling).

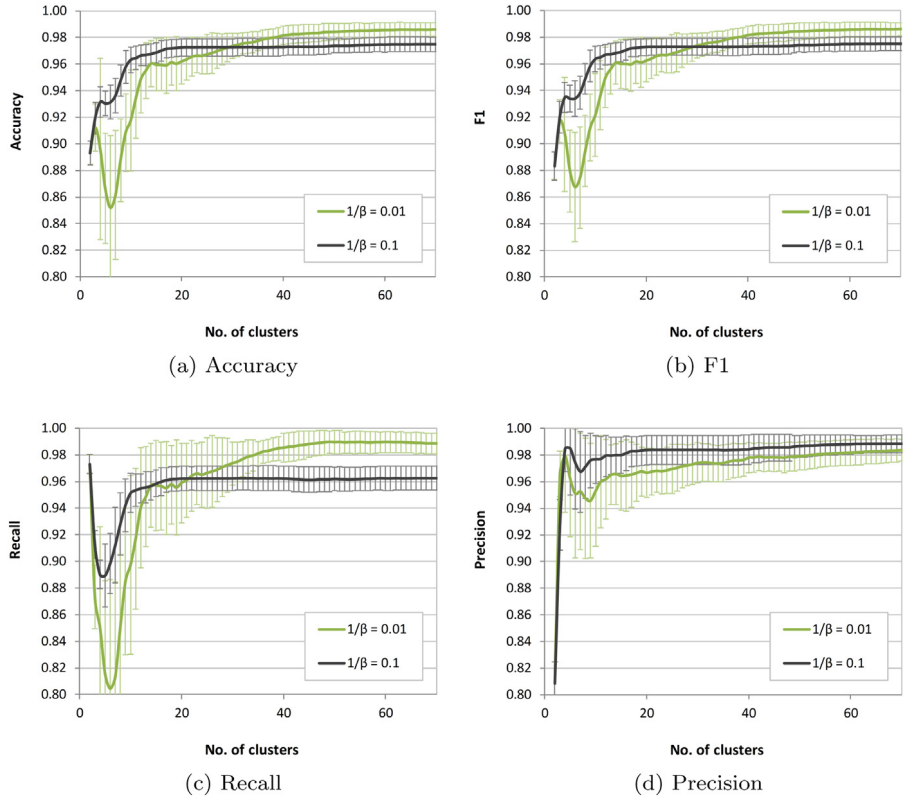

Fig. 13. Classification performance scores (aIB, majority class cluster labeling).

to the *majority class* labeling, without taking into account proportions of bots and humans in the mixed clusters, which may be very differentiated (like 10% to 90%) or equable (even 50% to 50%). To deal with this problem, in the second experimental scenario the threshold-based cluster labeling was applied, leading to a three-state classifier.

5.5.2. Three-state classifier

According to the threshold-based labeling procedure, for a given threshold $T_{mix} = X$ all clusters containing more than X% of majority class sessions were assigned a majority class label (*0* or *1*) and other clusters were labeled as $mixed_X$. This allowed us to introduce a three-state session classifier, with three possible outputs: *1* (bot), *0* (human), and $mixed_X$ (an *uncertain* session for $T_{mix} = X$). In this way we can mark the most ambiguous clusters and filter out uncertain, possibly suspicious or untypical user sessions.

We present classification results for two threshold values, 90 and 80. $T_{mix} = 90$ means that ambiguous clusters will be the ones having less than 90% of majority class session, i.e., characterized by entropy higher than 0.47. For the example visualized in Fig. 9, $mixed_{90}$ clusters are the following: for $k = 2, 3, 4$ this is cluster no. 1, containing 250 (89.9%) human sessions and 28 (10.1%) robot sessions (the cluster entropy is 0.471); for $k = 5, 6$ an ambiguous cluster is the one no. 2, containing 14 (34.1%) and 27 (65.9%) human and robot sessions, respectively (the cluster entropy is 0.926). When $T_{mix} = 80$, $mixed_{80}$ clusters will be the ones with the entropy higher than 0.72. For Fig. 9 example, a session partition for $k = 2, 3, 4$ does not contain any $mixed_{80}$ cluster and for $k = 5, 6$ this is cluster no. 2.

Fig. 14 shows performance scores for the three-state classifier in comparison with the scores for a binary classifier ($\beta^{-1} = 0.1$). As one might expect, results are better for higher T_{mix}. After filtering out most uncertain sessions, classifier efficiency greatly increases: when $T_{mix} = 90$, accuracy, F1, and recall stabilize at the level of about 0.99 just for $k = 10$. It means that about 99% of all sessions are correctly classified and 99% of bots are properly identified. Precision for ten clusters is also about 99%, which means that 99% of all robot classifications were correct and only 1% of human sessions were misclassified.

For the three-state classifier some sessions remain unclassified. Fig. 15 illustrates an impact of T_{mix} on the percentage of uncertain sessions as a function of the number of clusters. For $k < 4$ this percentage is unacceptably high but for higher k it drops rapidly and is more or less stable for about 18 clusters. For ten clusters there are 10.5% of uncertain sessions for $T_{mix} = 90$ and 6.7% for $T_{mix} = 80$, which seems to be an acceptable cost for a great improvement in classification quality. In practice, a website administrator may determine the parameter values based on his/her experience and the site specificity or may dynamically tune them to adapt a way of processing active sessions on a server to actual traffic, in accordance with the adopted security policy.

Distinguishing the most ambiguous clusters and identification of uncertain sessions is especially beneficial in terms of possible identification of malicious robots. Mixed clusters contain sessions which are difficult to clearly classify and it is very probable that some of them are robots impersonating humans.

5.5.3. Comparison of aIB with other clustering methods

Taking into consideration our research objectives, the proposed approach requires calculation of several fitness functions with the number of clusters given as an argument. This implies selection of a hierarchical clustering algorithm as a natural choice. We decided to apply the agglomerative IB method with β parameter which allows one to control the algorithm's "greed". This appears to be an important feature since the best performing value ranges of β turned out to be different for small and large numbers of clusters. To illustrate robustness of aIB in separating bots from humans of different types, two other agglomerative clustering techniques were implemented and verified: complete-linkage and Ward.D2 [42]; both based on a commonly applied Lance-Williams dissimilarity update formula.

Moreover, for the completeness of our work, we decided to compare the efficiency of aIB and some other, non-agglomerative clustering algorithms. There are a number of effective state-of-the-art clustering methods, including a new simplex sparse learning model [43], robust graph learning from noisy data [44], low-rank kernel learning for graph-based clustering [45], variance reduced k-means clustering [46], or clustering with similarity preserving [47]. These methods, however, are particularly effective for classification tasks with feature-rich, noisy data and a fixed range of small numbers of clusters. In contrast, one of our goal was reducing the number of session features (with aIB it was reduced to just six). Thus, we decided to select two well-known clustering algorithms, k-means [48] and NMF [49,50]. Even if they are not fully compatible with the hierarchical clustering concept, they have been widely applied in classification tasks for benchmarking purposes [43].

Fig. 16 juxtaposes clustering entropy rates achieved for five techniques: aIB (with $\beta^{-1} = 0.1$ and 0.01), k-means, NMF, Ward.D2, and complete-linkage, as a function of the number of clusters. We can see that when the number of subgroups is small, entropy for the reference techniques is much worse than for aIB (both with $\beta^{-1} = 0.1$ and 0.01); it improves with the increase in the number of subgroups in all cases. For small k the clustering results are especially poor for the reference hierarchical algorithms. As the number of clusters increases, the results of all algorithms except NMF tend to converge. However, taking into consideration that the value of β^{-1} may be modified depending on k, aIB is clearly most effective, achieving significantly lower entropy rates.

It has to be noted that k-means and NMF (as well as some of other state-of-the-art clustering methods mentioned above) have an additional drawback to rely on random initialization. We have investigated this problem in relation with our task for these two techniques. Minimum, maximum and pooled standard deviation was calculated for the entropy scores. The values of pooled standard deviation turned out to be maximal for three and four clusters — they were almost the same for both techniques and were close to 0.1. As the entropy for these cases is in the range 0.3–0.5, the pooled standard deviation is as high as 20%–30% of it. On the contrary, the aIB method is fully deterministic so it does not have this drawback.

This comparative part of our study can be concluded that the proposed approach based on aIB definitely surpasses the ones based on other clustering algorithms regardless of k, provided that the appropriate value of β^{-1} is selected depending on the number of clusters, as discussed earlier.

6. Conclusion

To date, many studies investigated differences between the online navigation of legitimate users and Web bots, various solutions for bot detection were proposed as well. Very few previous studies, however, investigated possibilities of discovering different profiles of bots and humans by unsupervised learning. We addressed this problem by a novel approach to develop a user session representation and a corresponding classification model. A session feature ranking was created by the Fisher Score algorithm and a subset of the most relevant features was selected based on experimentally-driven clustering entropy rates. The agglomerative Information Bottleneck algorithm was applied to cluster user sessions.

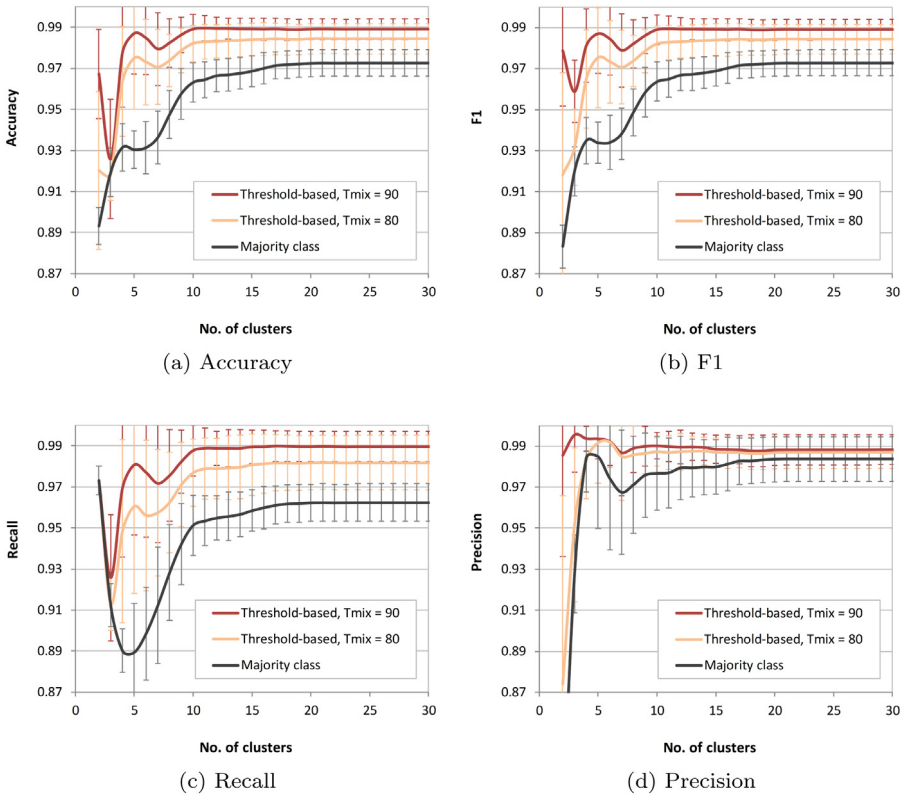

Fig. 14. Classification performance scores (aIB, threshold-based cluster labeling) vs.majority class labeling, $\beta^{-1} = 0.1$.

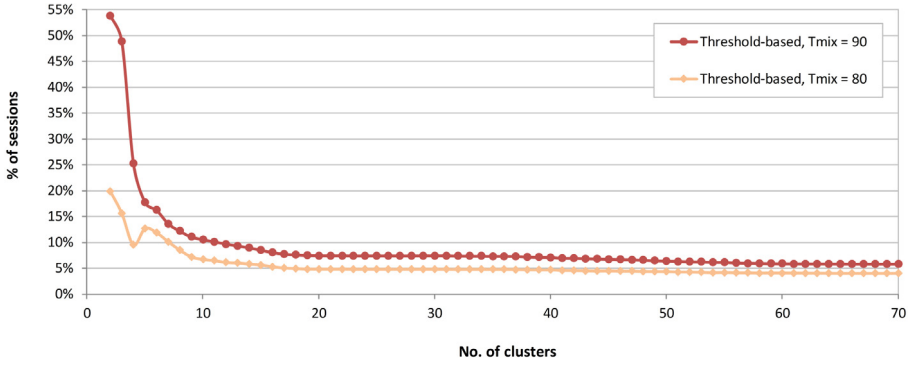

Fig. 15. Average percentage of sessions filtered out into the ambiguous session group (aIB, threshold-based cluster labeling, $\beta^{-1} = 0.1$).

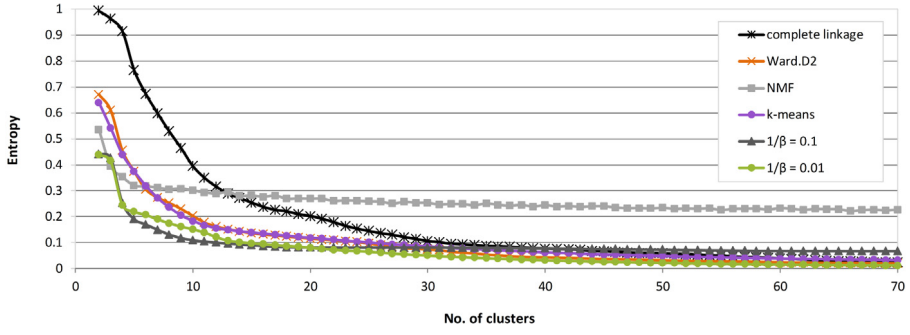

Fig. 16. Clustering entropy for aIB (majority class cluster labeling) in comparison with other clustering algorithms.

Performance of the proposed solution was evaluated experimentally using real e-commerce log data. An extensive session feature set was reduced to only six features. The aIB clustering, applied for a range of subset numbers, allowed us to distinguish multiple session profiles, giving a mean clustering entropy of 0.1 just for 18 clusters. Our findings show that legitimate users

exhibit similar online behaviors and thus, their sessions are partitioned into a very small number of subgroups. This is justified by the way of the human–website interaction, which proceeds according to the structure of available hyperlinks, determined by the Web interface. In contrast, robots reveal a variety of navigational patterns and their sessions are spread across multiple clusters.

Our clustering analysis showed that in the case of the relatively small number of subgroups, possible to be characterized by a human expert, an additional indication of a subgroup type (profile) may be deduced. This information may be used in practice by a website administrator to apply specific filters for session processing depending on the site specificity and the resource availability policy.

Moreover, clustering allowed us to identify some mixed clusters, which have a great chance to contain suspicious sessions of bots impersonating humans. In practice, a special attention should be payed to these sessions. Filtering out the most ambiguous instances gives the possibility to enforce special treatment for sessions of that type. They may be manually analyzed in detail or processed on a server in a special way, e.g., be subjected to further processing by other methods or augmented with a CAPTCHA authentication test to verify user identity.

A classification model was used to categorize new sessions. Two classifiers were proposed: a binary classifier, based on the majority class cluster labeling, and a three-state classifier, based on a threshold-based labeling. The binary classifier achieved very good results for higher numbers of clusters: for 50 subgroups it was able to correctly classify about 98% of sessions and detect almost 99% of robots on average; however, performance scores were much lower for small number of subgroups. Introducing the three state classifier allowed us to distinguish a great majority (about 90%) of "clear" bot or human sessions (assigned into pure clusters) and the remaining uncertain ones (allocated into the most ambiguous mixed clusters). This caused a great increase in classification performance for "pure" sessions even for small numbers of clusters: for ten subgroups mean values of accuracy, F1, recall, and precision of about 99% were achieved. Another benefit of this approach was gaining an ability to filter out the most uncertain sessions, difficult to interpret and classify.

The proposed solution can rely on a relatively small session set, containing 500 samples and its performance scores are quite stable. This indicates a possibility of its use on a real server, where due to the evolving nature of Web traffic, a classification model should be periodically updated.

In future work we are planning to extend our approach to identify malicious robots, also by applying a multinomial classification. Another research direction is adapting the solution to a real-time scenario, by updating descriptions of active sessions upon arrival of subsequent requests and trying to take a classification decision before the session ends. Such a solution would allow developing and enforcing differentiated session processing on a server depending on an identified profile. For example, malicious bots might be blocked and only human users might receive a personalized service (like personalized product recommendations), which would allow the better and safer use of server resources. Lastly, we will strive for verifying efficiency of our approach for other server log data to generalize results and investigate limitations and practical implications thoroughly.

Our methodology may be used to analyze user sessions on other websites, also in combination with different ground-truth session labeling policies, e.g., robots might be marked as benign and malicious ones or as known bots (revealing their identities in user agents or accessing *robots.txt* file) and camouflaged ones. Anyway, the study is a step towards developing more efficient bot detection strategies and ensuring more secure and reliable Web services.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

References

[1] GlobalDots, 2018 Bad Bot Report, Tech. rep., 2018, www.globaldots.com.
[2] Imperva Incapsula, Bot Traffic Report 2016, Tech. rep., 2017, https://www.incapsula.com/blog/bot-traffic-report-2016.html.
[3] S. Rovetta, G. Suchacka, F. Masulli, Bot recognition in a Web store: an approach based on unsupervised learning, J. Netw. Comput. Appl. 157 (2019) 102577, http://dx.doi.org/10.1016/j.jnca.2020.102577.
[4] D. Stevanovic, N. Vlajic, A. An, Detection of malicious and non-malicious website visitors using unsupervised neural network learning, Appl. Soft Comput. 13 (1) (2013) 698–708, http://dx.doi.org/10.1016/j.asoc.2012.08.028.
[5] K.R. Fall, W.R. Stevens, TCP/IP Illustrated, Addison–Wesley, 2011.
[6] F. Asdaghi, A. Soleimani, An effective feature selection method for Web spam detection, Knowl.-Based Syst. 166 (2019) 198–206, http://dx.doi.org/10.1016/j.knosys.2018.12.026.
[7] D. Yu, N. Chen, F. Jiang, B. Fu, A. Qin, Constrained NMF-based semi-supervised learning for social media spammer detection, Knowl.-Based Syst. 125 (2017) 64–73, http://dx.doi.org/10.1016/j.knosys.2017.03.025.
[8] D. Acarali, M. Rajarajan, N. Komninos, I. Herwono, Survey of approaches and features for the identification of HTTP-based botnet traffic, J. Netw. Comput. Appl. 76 (2016) 1–15, http://dx.doi.org/10.1016/j.jnca.2016.10.007.
[9] S. Lysenko, K. Bobrovnikova, O. Savenko, A. Kryshchuk, BotGRABBER: SVM-based self-adaptive system for the network resilience against the botnets' cyberattacks, in: International Conference on Computer Networks, Springer, 2019, pp. 127–143.
[10] K.L. Chiew, K.S.C. Yong, C.L. Tan, A survey of phishing attacks: Their types, vectors and technical approaches, Expert Syst. Appl. 106 (2018) 1–20, http://dx.doi.org/10.1016/j.eswa.2018.03.050.
[11] H. Cai, F. Zhang, Detecting shilling attacks in recommender systems based on analysis of user rating behavior, Knowl.-Based Syst. 177 (2019) 22–43, http://dx.doi.org/10.1016/j.knosys.2019.04.001.
[12] W. Guo, S. Ju, Y. Gu, Web robot detection techniques based on statistics of their requested URL resources, in: Proceedings of the 9th International Conference on Computer Supported Cooperative Work in Design, Vol. 1, 2005, pp. 302–306, http://dx.doi.org/10.1109/CSCWD.2005.194187.
[13] S. Kwon, M. Oh, D. Kim, J. Lee, Y.-G. Kim, S. Cha, Web robot detection based on monotonous behavior, in: Proceedings of the Information Science and Industrial Applications, Vol. 4, 2012, pp. 43–48, http://dx.doi.org/10.1109/CSCWD.2005.194187.
[14] X. Lin, L. Quan, H. Wu, An automatic scheme to categorize user sessions in modern HTTP traffic, in: Proc. IEEE GLOBECOM'08, 2008, pp. 1–6, http://dx.doi.org/10.1109/GLOCOM.2008.ECP.290.
[15] D. Doran, S.S. Gokhale, An integrated method for real time and offline Web robot detection, Expert Syst. 33 (6) (2016) 592–606, http://dx.doi.org/10.1111/exsy.12184.
[16] S. Kwon, Y.-G. Kim, S. Cha, Web robot detection based on pattern-matching technique, J. Inf. Sci. 38 (2) (2012) 118–126, http://dx.doi.org/10.1177/0165551511435969.
[17] W.-Z. Lu, S.-Z. Yu, Web robot detection based on hidden Markov model, in: Proceedings of International Conference on Communications, Circuits and Systems, Vol. 3, 2006, pp. 1806–1810, http://dx.doi.org/10.1109/ICCCAS.2006.285024.
[18] A. Stassopoulou, M.D. Dikaiakos, Web robot detection: a probabilistic reasoning approach, Comput. Netw. 53 (3) (2009) 265–278, http://dx.doi.org/10.1016/j.comnet.2008.09.021.
[19] G. Suchacka, M. Sobków, Detection of internet robots using a Bayesian approach, in: Proceedings of IEEE 2nd International Conference on Cybernetics, 2015, pp. 365–370, http://dx.doi.org/10.1109/CYBConf.2015.7175961.
[20] A. Balla, A. Stassopoulou, M.D. Dikaiakos, Real-time Web crawler detection, in: Proc. Int. Conf. Telecommunications, 2011, pp. 428–432, http://dx.doi.org/10.1109/CTS.2011.5898963.
[21] C. Bomhardt, W. Gaul, L. Schmidt-Thieme, Web robot detection - preprocessing Web logfiles for robot detection, in: New Developments in Classification and Data Analysis, Springer, Berlin, Heidelberg, 2005, pp. 113–124.

[22] T. Gržinić, L. Mršić, J. Šaban, Lino - an intelligent system for detecting malicious Web-robots, in: Intelligent Information and Database Systems. ACIIDS'15, in: LNAI, vol. 9012, Springer International Publishing, Cham, 2015, pp. 559–568, http://dx.doi.org/10.1007/978-3-319-15705-4_54.
[23] G. Jacob, E. Kirda, C. Kruegel, G. Vigna, PUBCRAWL: Protecting users and businesses from CRAWLers, in: Proceedings of the 21st USENIX Security Symposium, Security'12, USENIX Association, Berkeley, CA, USA, 2012, p. 25.
[24] S. Rovetta, A. Cabri, F. Masulli, G. Suchacka, Bot or not? A case study on bot recognition from Web session logs, in: Quantifying and Processing Biomedical and Behavioral Signals, in: Smart Innovation, Systems and Technologies, vol. 103, Springer, 2019, pp. 197–206.
[25] C.H. Saputra, E. Adi, S. Revina, Comparison of classification algorithms to tell bots and humans apart, J. Next Gen. Inf. Technol. 4 (7) (2013) 23–32.
[26] D.S. Sisodia, S. Verma, O.P. Vyas, Agglomerative approach for identification and elimination of Web robots from Web server logs to extract knowledge about actual visitors, J. Data Anal. Inf. Process. 03 (2015) 1–10, http://dx.doi.org/10.4236/jdaip.2015.31001.
[27] D. Stevanovic, A. An, N. Vlajic, Feature evaluation for Web crawler detection with data mining techniques, Expert Syst. Appl. 39 (10) (2012) 8707–8717, http://dx.doi.org/10.1016/j.eswa.2012.01.210.
[28] P.-N. Tan, V. Kumar, Discovery of Web robot sessions based on their navigational patterns, Data Min. Knowl. Discov. 6 (1) (2002) 9–35, http://dx.doi.org/10.1023/A:1013228602957.
[29] J. Hamidzadeh, M. Zabihimayvan, R. Sadeghi, Detection of Web site visitors based on fuzzy rough sets, Soft Comput. 22 (7) (2018) 2175–2188, http://dx.doi.org/10.1007/s00500-016-2476-4.
[30] G. Suchacka, Improving clustering of Web bot and human sessions by applying Principal Component Analysis, in: Proceedings of the 33rd International ECMS Conference on Modelling and Simulation (ECMS'19), 2019, pp. 000–000.
[31] M. Zabihi, M.V. Jahan, J. Hamidzadeh, A density based clustering approach to distinguish between Web robot and human requests to a Web server, ISC Int. J. Inf. Secur. 6 (1) (2014) 77–89.
[32] M. Zabihimayvan, R. Sadeghi, H.N. Rude, D. Doran, A soft computing approach for benign and malicious Web robot detection, Expert Syst. Appl. 87 (2017) 129–140, http://dx.doi.org/10.1016/j.eswa.2017.06.004.
[33] Udger, https://udger.com. (Accessed 4 September 2017).
[34] User-agents, https://http://www.user-agents.org. (Accessed 4 September 2017).
[35] G. Suchacka, I. Motyka, Efficiency analysis of resource request patterns in classification of Web robots and humans, in: Proceedings of the 32nd European Conference on Modelling and Simulation, 2018, pp. 475–481, http://dx.doi.org/10.1109/CYBConf.2015.7175961.
[36] J. Pohjalainen, O. Räsänen, S. Kadioglu, Feature selection methods and their combinations in high-dimensional classification of speaker likability, intelligibility and personality traits, Comput. Speech Lang. 29 (1) (2015) 145–171, http://dx.doi.org/10.1016/j.csl.2013.11.004.
[37] Q. Gu, Z. Li, J. Han, Generalized fisher score for feature selection, in: Proceedings of the 27th Conference on Uncertainty in Artificial Intelligence, UAI'11, AUAI Press, Arlington, Virginia, United States, 2011, pp. 266–273.
[38] N. Tishby, F.C. Pereira, W. Bialek, The information bottleneck method, in: Proceedings of IEEE Congress on Evolutionary Computation, 1999, pp. 22–24.
[39] N. Slonim, N. Tishby, Agglomerative information bottleneck, Adv. Neural Inf. Process. Syst. 12 (1999) 617–623.
[40] N. Slonim, The information bottleneck: Theory and applications (Ph.D. thesis), Hebrew University, Jerusalem, Israel, 2002, http://www.cs.huji.ac.il/labs/learning/Theses/Slonim_PhD.pdf.
[41] P.-N. Tan, M. Steinbach, V. Kumar, Introduction to Data Mining, Pearson Addison-Wesley, Boston, MA, USA, 2006.
[42] F. Murtagh, P. Legendre, Ward's hierarchical agglomerative clustering method: Which algorithms implement Ward's criterion? J. Classification 31 (3) (2014) 274–295, http://dx.doi.org/10.1007/s00357-014-9161-z.
[43] J. Huang, F. Nie, H. Huang, A new simplex sparse learning model to measure data similarity for clustering, in: 24th International Joint Conference on Artificial Intelligence, IJCAI'15, AAAI Press, 2015, pp. 3569–3575.
[44] Z. Kang, H. Pan, S.C.H. Hoi, Z. Xu, Robust graph learning from noisy data, IEEE Trans. Cybern. (2019) 1–11, http://dx.doi.org/10.1109/TCYB.2018.2887094.
[45] Z. Kang, L. Wen, W. Chen, Z. Xu, Low-rank kernel learning for graph-based clustering, Knowl.-Based Syst. 163 (2019) 510–517, http://dx.doi.org/10.1016/j.knosys.2018.09.009.
[46] Y. Zhao, Y. Ming, X. Liu, E. Zhu, J. Yin, Variance reduced k-means clustering, in: The 32nd AAAI Conference on Artificial Intelligence, AAAI'18, AAAI Press, 2018, pp. 8187–8188.
[47] Z. Kang, H. Xu, B. Wang, H. Zhu, Z. Xu, Clustering with similarity preserving, 2019, CoRR abs/1905.08419, arXiv:1905.08419.
[48] J. MacQueen, et al., Some methods for classification and analysis of multivariate observations, in: Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability, Vol. 1, 1967, pp. 281–297.
[49] J.-P. Brunet, P. Tamayo, T.R. Golub, J.P. Mesirov, Metagenes and molecular pattern discovery using matrix factorization, Proc. Natl. Acad. Sci. USA 101 (12) (2004) 4164–4169, http://dx.doi.org/10.1073/pnas.0308531101.
[50] R. Gaujoux, C. Seoighe, A flexible R package for nonnegative matrix factorization, BMC Bioinform. 11 (1) (2010) 367, http://dx.doi.org/10.1186/1471-2105-11-367.