A Comparative Study of Negative Selection Based Anomaly Detection in Sequence Data

Johannes Textor

Theoretical Biology & Bioinformatics
Universiteit Utrecht
Paudalaan 8
3584 CH Utrecht, The Netherlands
johannes.textor@gmx.de

Abstract. The negative selection algorithm is one of the oldest immune-inspired classification algorithms and was originally intended for anomaly detection tasks in computer security. After initial enthusiasm, performance problems with the algorithm lead many researchers to conclude that negative selection is not a competitive anomaly detection technique. However, in recent years, theoretical work has lead to substantially more efficient negative selection algorithms. Here, we report the results of the first evaluation of negative selection with r-chunk and r-contiguous detectors that employs these novel algorithms. On a collection of 14 datasets from real-world sources, we compare negative selection with r-chunk and r-contiguous detectors against techniques based on kernels, finite state automata, and n-gram frequencies, and find that negative selection performs competitively, yielding a slightly better average performance than all other techniques investigated. Because this study represents, to our knowledge, the most comprehensive one of string-based negative selection to date, the widely held view that negative selection is not a competitive anomaly detection technique may be inaccurate.

1 Introduction

One of the first immunological paradigms to be used as an inspiration for algorithm design was the paradigm of *negative selection*. In brief, the so-called *T-cells*, immune cells bearing diverse receptors highly specialized to recognizing specific molecular patterns, are born in the bone marrow and initially carry receptors generated by random recombination of DNA fragments. These T-cells are then exposed to normal proteins from the host organism (self) in an organ called the thymus. T-cells which react to such self proteins are eliminated, and only those that survive this process finally egress from the thymus and become part of the immune system.

Forrest et al. [1] abstracted this immunological principle into a simple anomaly detection algorithm: Based on pre-defined notions of a universe $\mathcal{U}$ and a set of patterns (detectors) $\mathcal{P}$, which each match a small subset of the universe, the algorithm first generates a set P of patterns that all fail to match the elements

of an input dataset $S \subseteq \mathcal{U}$ by rejection sampling. Then, each elements of another input dataset X is classified as anomalous if it is matched by any of the patterns in P and as normal, otherwise.

Initially researchers were enthusiastic about negative selection [2], however, performance problems soon became obvious [3]: For many datasets, the number of patterns necessary to achieve a good sensitivity rate is prohibitively large [4], while for others, it can take exponential time for rejection sampling to find even a single detector [5]. For these reasons, limited work has been done on evaluating especially string-based negative selection on real-world datasets, and such studies (e.g. [6], [7]) often lack a comparison to more standard anomaly detection techniques. In the case of string based negative selection, the only exception, to our knowledge, is the work of Stibor, who analyzed the performance of negative selection algorithms on four artificially generated binary datasets [8] and, in later work, compared the performance to Kernel density estimation [9].

In more recent work, Elberfeld and Textor [10, 11] showed that the complexity issues pointed out by Stibor [12] can be solved, and negative selection with both r-chunk and r-contiguous patterns can be implemented in polynomial time. The main trick behind their work is the use of data compression techniques to avoid generating exponentially large detector sets explicitly. Theoretically, this approach should allow processing of sizable string-based datasets in reasonable time, even for large alphabets. Here, we report, to our knowledge, the first empirical results using these new algorithms. Our study employs 14 sequence datasets consisting of hundreds to thousands of sequences with alphabet sizes of up to 78. The performance of the negative selection algorithms is compared to that of 5 state-of-the-art anomaly detection techniques shown to perform competitively on 10 of these datasets in earlier work by Chandola et al [13].

The structure of this paper is as follows. In the upcoming section, we define the kind of classification problem we are solving and introduce our evaluation methodology. In Section 3, we give a brief overview of the classification algorithms we are employing. However, because the techniques we employ in this paper have all been described in great detail elsewhere, we give only a very high-level overview and refer the interested reader to the literature. Section 4 then describes the datasets we are using. Our experimental results are presented in Section 5. The paper concludes with a discussion of these results (Section 6).

2 Preliminaries

2.1 Problem definition

Formally, the classification problem at hand can be defined as follows. Given an alphabet Σ, the input consists of two sets $S, X \subseteq \Sigma^*$. The set S is itself a subset of an unknown set (real self, ground truth) $\mathcal{S} \subseteq \Sigma^*$, and another set $X \subseteq \Sigma^*$. The algorithm's task is to decide, based on the information in S, for each $x \in X$ whether $x \in \mathcal{S}$ holds. The algorithms considered in this paper will do so by assigning to each $x \in X$ a value $\alpha(x) \in \mathbb{R}$ (anomaly score), quantifying

the algorithm's confidence that x is an anomaly (i.e., $x \notin \mathcal{S}$). This type of a classification problem is usually referred to as "one-class classification" [14] and can also be viewed as a kind of semi-supervised learning [15, 16]. In the following, S will also be referred to as "training data" and X as "test data".

Some of the algorithms we use, including negative selection, actually take strings of a fixed length ℓ as input. In that case, we transform the input dataset S into another set S^ℓ containing all substrings of length ℓ of all strings in S. To assign an anomaly score to a sequence $x \in X$, we first compute the anomaly score for each substring of length ℓ of x, giving a vector $(\alpha_1, \alpha_2, \ldots, \alpha_{|x|-\ell+1})$ of scores. Then, the final anomaly score α is defined as the logsum of the individual scores, i.e.

$$\alpha = \sum_i \log \alpha_i \ . \tag{1}$$

If some of the α_i are 0, we replace them with 10^{-6}. Of course, many other ways are conceivable to combine the sliding window anomaly scores into a single score [13]. We use the logsum here because it has been shown to perform well for the techniques studied by Chandola et al [13].

2.2 Evaluation Methodology

The results of our analysis were evaluated quantitatively as follows. First, like in Chandola et al. [13], we ranked all test sequences by their anomaly score and counted the percentage of true anomalies among the top n sequences, where n is the number of true anomalies in the test dataset X. Hence, a value of 1.0 would mean that the anomalous sequences are perfectly separated from the normal sequences, while a value of 0.0 would mean that the top n sequences are all false positives.

As a second evaluation metric, we used a receiver operating characteristic (ROC) analysis. A ROC curve visualizes the trade-off between sensitivity and specificity: For a given threshold θ for the anomaly score, one determines the false positive rate

$$\mathrm{FP}_\theta = \frac{\#\text{ normal instances scoring higher than } \theta}{\#\text{ normal instances}} \tag{2}$$

and the true positive rate

$$\mathrm{TP}_\theta = \frac{\#\text{ anomalous instances scoring higher than } \theta}{\#\text{ anomalous instances}} \ . \tag{3}$$

The ROC curve is then given by the points $(\mathrm{FP}_\theta, \mathrm{TP}_\theta)$ for all possible values of θ. The larger the area under the ROC curve (AUC), the better the performance of the classification algorithm. The ROC curve of a "classifier" that tosses a coin to determine the label of each $x \in X$ is a diagonal line from the origin to the point $(1, 1)$ with an AUC of 0.5. The ROC curve of a meaningful classifier should thus lie well above the diagonal and have an AUC higher than 0.5. A near-perfect

classifier, which assigns almost all labels correctly for almost all parameters, has an AUC close to 1.

To compare the two metrics, we note that for our dataset, the percentage of true anomalies near the top is a "tougher" measure because our datasets contain only few anomalies. This means that a classifier which positions all anomalies near the top, with a few false positives in between, will get a very good AUC (near 1) but may still perform poorly in the true positive rate metric.

3 Algorithms

We start by briefly re-stating the definitions of negative selection algorithms with r-chunk and r-contiguous patterns (detectors). After that, we briefly explain the algorithms used by Chandola et al. [13], and refer the reader to that source for a more complete explanation. Note that Chandola et al. originally analyzed 7 different algorithms. However, two of those were shown to perform much worse than the other five on almost all datasets. Therefore, we excluded those algorithms from our analysis to obtain a more competitive reference base for our benchmark.

3.1 Negative Selection

Given are a universe $\mathcal{U}$, a pattern set $\mathcal{P}$ and a matching function that, for every $\pi \in \mathcal{P}$ and every $x \in \mathcal{U}$, determines whether π matches x or not. For any $S \subseteq \mathcal{U}$, let $\mathcal{P}[S]$ denote the set of patterns that do not match any $s \in S$, called the set of *S-consistent patterns* [17].

Definition 1 (r-chunk pattern). *An r-chunk pattern (d, i) is a tuple of a string $d \in \Sigma^r$ and an integer $i \in \mathbb{N}$. It matches a string $s \in \Sigma^\ell$ if $i \in \{1, \ldots, \ell - r + 1\}$ and the substring of s of length r starting at the i-th position is equal to d.*

When using negative selection with r-chunk patterns, we proceed as follows. For the input S, first generate $\mathcal{P}[S]$. Then, for each $x \in X$, the anomaly score is defined as the number of elements of $\mathcal{P}[S]$ that also match x. Hence, anomaly scores range from 0 to $\ell - r + 1$. For computing the logsum, we add 1 to each anomaly score.

Definition 2 (r-contiguous patterns). *An r-contiguous pattern is a string $d \in \Sigma^\ell$. It matches another string $s \in \Sigma^\ell$ if d and s are identical in at least r contiguous positions, i.e., if there is an $i \in \{1, \ldots, l - r + 1\}$ such that the substrings of length r of s and d starting at the i-th position are equal.*

For negative selection with r-contiguous patterns, we also first generate the set $\mathcal{P}[S]$ of S-consistent patterns. Then, for each $x \in X$, the anomaly score is defined as the largest number $r' \in \{0, \ldots, \ell\}$ such that there exists a pattern $\pi \in \mathcal{P}[S]$ that is identical to x in r' contiguous positions. Hence, anomaly scores

range from 0 (we define that two strings are always identical in 0 contiguous positions) to ℓ. Again, we add 1 to all anomaly scores for computing the logsum.

Note that this type of anomaly score is somewhat different to what would typically be used in negative selection with r-contiguous detectors. Classically, one would simply determine whether or not there exists an S-consistent r-contiguous pattern that matches x. Our method generalizes this procedure in the following sense: If an r-contiguous pattern exists, then the anomaly score r' will be greater than or equal to r, otherwise it will be smaller than r.

We note that while the above explanations serve as a simplified description of our algorithms, we did in fact not generate the entire sets of S-consistent patterns, as these would have been prohibitively large, especially for the larger alphabets. Instead, we used the algorithms presented in earlier work by Elberfeld and Textor [11] and Liśkiewicz and Textor [17], which generate compressed representations of these sets. Further, note that Liśkiewicz and Textor [17] also presented an efficient algorithm to count the S-consistent r-contiguous patterns that match x, a number which they termed the *detector sampling distance*. We implemented this algorithm and found it to perform in almost all cases exactly like the corresponding r-chunk-based algorithm explained above. For this reason, we decided to use instead the simpler anomaly score for r- contiguous patterns discussed above. Our reference implementation (see end of this paper), however, contains both anomaly score versions as a basis for future work.

3.2 Algorithms based on ℓ-grams

Like negative selection, the three algorithms of this group are applied by computing the logsum of anomaly scores for a sliding window of fixed size ℓ (see previous section). The easiest technique is the t-STIDE technique (stide) [18]. Here, the anomaly score is defined as

$$\alpha(s) = \frac{f(s)}{f(*)} \tag{4}$$

where $f(s)$ is the number of times that the string s occurs in the training data S, and $f(*)$ is the total number of strings of length ℓ in S. Note that for this score, higher numbers mean less anomalous sequences, which is also true for the following two techniques.

The other two algorithms in this group are based on finite state automata (FSA), which compute the conditional probability of the last symbol in the string given the $\ell - 1$ symbols preceding it. For instance, if the sequence AAAAA was followed most frequently by the letter B in the training data, then an occurrence of AAAAAB in the test data would get a high likelihood score and an occurrence of AAAAAC would get a low likelihood score. We refer to the literature for a detailed description on how this automaton is constructed [19]. The basic version of the algorithm (fsa) processes only windows from the test data that also occurred in the training data. Our third ℓ-gram based technique, FSA-z (fsaz), is a slight extension of FSA which assigns a likelihood score of 0 to unseen sequences instead of ignoring them.

3.3 Kernel based algorithms

In contrast to negative selection and ℓ-gram based techniques, kernel based algorithms treat the sequences as a whole instead of using a sliding window. Let $u, v \in \Sigma^*$ be two strings, then the kernel function $\kappa(u, v) \in [0, 1]$ representing their similarity is defined as follows:

$$\kappa(u, v) = \frac{|\mathrm{LCS}(u, v)|}{\sqrt{|u|\,|v|}} \ , \tag{5}$$

where $\mathrm{LCS}(u, v)$ denotes the longest common substring (not necessarily starting at the same position) of u and v.

Based on this kernel function, we use the following two classifiers: (1) A simple kNN classifier (knn), where the anomaly score is the distance to the kth nearest neighbour in the training set; (2) a clustering algorithm (cls), which partitions the training data into k clusters using the CLARA k-medoid algorithm [20].

Of course, there exist a plethora of other possibilities to define string similarity functions for kernel classifiers, which we do not yet explore here. We limit ourselves to the longest common substring based kernel for the simple reason that it was previously shown to perform competitively on 10 of the 14 datasets that we will analyze [13].

4 Datasets

class	dataset	$\lvert\Sigma\rvert$	$\langle\ell\rangle$	$\lvert S\rvert$	$\lvert X_{\mathrm{normal}}\rvert$	$\lvert X_{\mathrm{anomalous}}\rvert$
languages	hiligaynon	27	10	403	495	55
	latin	27	10	403	495	55
	plautdietsch	27	10	403	495	55
	tagalog	27	10	403	495	55
proteins	hcv	44	87	1423	1000	50
	nad	42	160	1685	1000	50
	tet	42	52	952	1000	50
	rub	42	182	559	500	25
	rvp	46	95	935	1000	50
syscalls	snd-cert	56	803	811	1000	50
	snd-unm	53	839	1030	1000	50
	bsm-week1	67	149	10	1000	50
	bsm-week2	73	141	113	200	10
	bsm-week3	78	143	67	1000	50

Table 1. Properties of the datasets used in our experimental evaluation. For each dataset, we list the alphabet size $|\Sigma|$, the mean length of sequences $\langle\ell\rangle$, as well as the sizes of the sets of training sequences (S), normal test sequences (X_{normal}), and anomalous test sequences ($X_{\mathrm{anomalous}}$).

We evaluated all algorithms on 14 datasets in total, out of which we generated the first 4 ourselves, while the latter 10 datasets were taken from the work of Chandola et al [13]. The ratio of anomalous to normal sequences in all test dataset ranges between 1:10 and 1:20. Table 1 shows the basic properties of all datasets.

We point out that our evaluation method is a deterministic one, and that the training dataset S (which contain only normal sequences) and test dataset X (which contain mostly normal and some anomalous sequences in the ratio shown in Table 1) are pre-defined in each case. Therefore, the performance of an algorithm on a dataset $S, X.$ is determined by a single run of the algorithm on that dataset. This is different from probabilistic evaluation methods like cross-validation, where training and test data are generated at random by splitting a basis dataset in two randomly chosen parts.

4.1 Languages

First we were interested how the algorithms would perform on a rather routine task, i.e., detection of short foreign-language snippets in an English-language background. In these datasets, all training and test sequences are of length 10. As training data, we extracted non-overlapping substrings of length 10 from the first two pages of Hermann Melville's classic novel "Moby Dick" [21], converting all letters to lower case, and collapsing all contiguous strings of non-letters to a single space. This yielded 403 strings for the input sample S. To generate normal test data, we applied the same procedure to the first chapter of the Book of John from the English Bible, giving 495 strings. As anomalous instances, we used the first 55 strings from the Book of John in the languages Hiligaynon (an austronesian language spoken in the Philippines), Latin, Plautdietsch (a northern German dialect), and Tagalog. Importantly, all these languages use the same alphabet – if this were not the case, then the anomaly detection problem would be rather trivial.

4.2 Protein families

For the protein family datasets, Chandola et al. [13] obtained sequences from the five protein families HCV, NAD, TET, RVP, and RUB from the PFAM database [22]. Proteins from different families are expected to structurally differ from one another, which is reflected in their amino acid sequences.

The sequences obtained for each family were split into training data and the normal part of the test data. The test datasets were then augmented by anomalous instances by sampling 50 sequences from the respective other families.

4.3 Intrusion detection

The intrusion detection datasets are of particular interest for our purpose, since the negative selection algorithm was originally conceived for intrusion detection,

and it has been claimed that negative selection is inappropriate for this task [8]. However, as mentioned at the beginning of this paper, this claim was in large parts based on now disproved speculations about the algorithm's performance, while no direct empirical evidence was given except some experiments on artificially constructed datasets which had no relation to intrusion detection and which were not sequence data in nature. Presumably, the performance issues of classical negative selection algorithms impeded a more thorough investigation.

The intrusion detection datasets consist of sequences of UNIX system calls made by processes operating either normally or being hacked / under attack, an idea put forward by Forrest et al [23]. Each symbol of the alphabet encodes one type of system call. Two of the datasets (snd-unm and snd-cert) were generated by that group at the University of New Mexico[1], while three others (bsm-week1 through bsm-week3) were collected from a Solaris machine at the DARPA Lincoln Labs [24].

5 Experimental Results

5.1 Tuning Procedure

It is most meaningful to compare the results of machine learning algorithms when these have undergone a similar degree of tuning. The negative selection algorithms both have two parameters that can be used to control generalization, namely n and r, while the other algorithms have only one parameter each. To enable a fair comparison, we therefore chose to set the parameter n to a value determined by the structure of the dataset, namely the length of the shortest string. This is the maximum value for n and therefore also leaves a maximum number of choices for the parameter r; however, note that the classification results could in principle still be better for smaller n. In this manner, we set $n = 10$ for the language datasets (which only consist of strings of length 10) and $n = 6$ for the other datasets. The negative selection algorithm was then tuned by modifying the parameter r only.

Tuning was then performed separately for each dataset class (languages, proteins, and system calls) by, for each algorithm, determining that parameter which gave the best average detection rate among the top n sequences (our first metric). This led to the parameter settings shown in Table 2. Note that for all datasets used by Chandola et al [13], we determined the same "optimal" parameter settings for the datasets they analyzed, except for kNN where we consistently found a slightly better performance with $k = 1$ than with the parameter $k = 4$ that they used[2]. Therefore, we here report the results for $k = 1$.

5.2 Performance

The numerical results of our experiments are shown in Table 3 through Table 5.

[1] http://www.cs.unm.edu/ immsec/systemcalls.html

[2] More specifically, Chandola et al [13] state that they find similar performance for $k \leq 4$ but find performance to become worse for larger k.

	languages	proteins	syscalls
fsa	$\ell = 5$	$\ell = 6$	$\ell = 6$
fsaz	$\ell = 5$	$\ell = 6$	$\ell = 6$
stide	$\ell = 5$	$\ell = 6$	$\ell = 6$
knn	$k = 1$	$k = 1$	$k = 1$
cls	$k = 22$	$k = 32$	$k = 32$
rchunk	$r = 7$	$r = 5$	$r = 5$
rcont	$r = 7$	$r = 5$	$r = 6$

Table 2. Parameter settings used to achieve the results shown in Table 3 through Table 5. For each technique, the parameters were tuned to yield the best average rate of true positives within the highest scoring input sequences.

Concerning the fraction of high-scoring true positives, the language datasets seem overall most difficult (perhaps due to their smallest size), followed by the anomaly detection datasets, while the protein family datasets seem easiest. Within the intrusion detection class, the DARPA datasets seem more difficult than the UNM datasets. All these observations are consistent with what was reported by Chandola et al [13]. The negative selection algorithms outperform all other techniques on the language datasets, come in second after kNN on the protein datasets, and score worst on the intrusion detection datasets where they come in 4th and 5th out of 7 techniques. A similar picture emerges when looking at the AUC metric, although the fraction of high-scoring true positives appears overall more informative for our benchmarking purposes.

The kernel based techniques exhibit the largest variation: They lag far behind on the language datasets, but are slightly ahead on the intrusion detection datasets, on which the negative selection algorithms show only average performance.

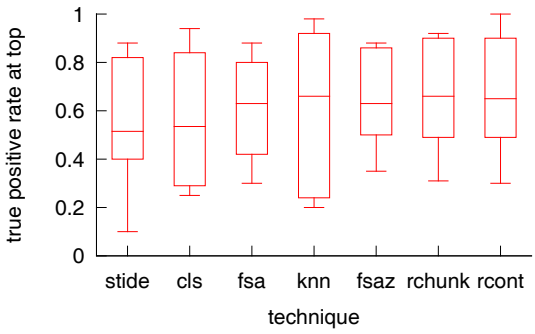

Fig. 1. Box plots depicting for each technique the minimum, maximum, interquartile range, and median across of the fraction of true anomalies among the top scoring test sequences for each of our 14 datasets (ordered by average performance).

1. Fraction of true anomalies

	fsa	fsaz	stide	knn	cls	rchunk	rcont	avg
hiligaynon	0.51	0.51	0.53	0.29	0.23	**0.59**	**0.59**	0.48
latin	**0.55**	0.51	0.49	0.25	0.23	0.44	0.44	0.41
plautdietsch	0.35	0.40	0.40	0.28	0.20	**0.47**	**0.47**	0.37
tagalog	0.35	0.35	0.40	0.28	0.24	**0.49**	**0.49**	0.38
avg	0.44	0.44	0.45	0.28	0.22	0.50	0.50	

2. Area under ROC curve

	fsa	fsaz	stide	knn	cls	rchunk	rcont	avg
hiligaynon	0.84	0.84	0.83	0.75	0.69	**0.92**	**0.92**	0.84
latin	0.85	0.85	0.85	0.72	0.67	**0.86**	**0.86**	0.81
plautdietsch	0.80	0.84	0.84	0.76	0.68	**0.90**	**0.90**	0.82
tagalog	0.79	0.80	0.81	0.73	0.70	**0.90**	**0.90**	0.81
avg	0.82	0.83	0.83	0.74	0.69	0.90	0.90	

Table 3. Results on language datasets. Above, bold numbers denote the best result in each row.

1. Fraction of true anomalies

	fsa	fsaz	stide	cls	knn	rchunk	rcont	avg
rvp	0.88	0.86	0.86	0.84	**0.96**	0.90	0.90	0.89
rub	0.88	0.88	0.88	0.72	**0.92**	**0.92**	**0.92**	0.87
nad	0.66	0.62	0.60	0.48	0.72	0.68	0.68	0.63
hcv	0.88	0.88	0.88	0.62	**0.98**	0.90	0.90	0.86
tet	0.72	0.82	0.82	**0.86**	0.84	0.78	0.78	0.80
	0.80	0.81	0.81	0.70	0.88	0.84	0.84	

2. Area under ROC curve

	fsa	fsaz	stide	cls	knn	rchunk	rcont	avg
rvp	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
rub	1.00	1.00	1.00	0.99	1.00	1.00	1.00	1.00
nad	0.97	0.97	0.96	0.95	0.98	0.96	0.96	0.96
hcv	1.00	1.00	1.00	0.98	1.00	1.00	1.00	1.00
tet	0.99	0.99	0.99	1.00	1.00	0.99	0.99	0.99
	0.99	0.99	0.99	0.98	1.00	0.99	0.99	

Table 4. Results on protein family datasets. Bold numbers denote the best result in each row. The best AUC values were not highlighted because they are too similar across the columns.

1. Fraction of true anomalies

	fsa	fsaz	stide	cls	knn	rchunk	rcont	avg
bsm-week1	0.30	**0.50**	0.10	**0.50**	0.40	0.31	0.30	0.34
bsm-week2	0.42	0.48	0.26	0.42	0.54	**0.56**	0.50	0.45
bsm-week3	0.60	**0.64**	0.54	0.57	0.60	**0.64**	0.62	0.60
snd-cert	0.78	0.88	0.50	0.94	0.94	0.90	**1.00**	0.85
snd-unm	0.80	0.76	0.46	0.84	**0.92**	0.72	0.72	0.75
	0.58	0.65	0.37	0.65	0.68	0.63	0.63	

2. Area under ROC curve

	fsa	fsaz	stide	cls	knn	rchunk	rcont	avg
bsm-week1	0.87	**0.90**	0.55	0.74	0.68	0.64	0.53	0.70
bsm-week2	0.87	0.89	0.68	**0.90**	0.86	0.77	0.79	0.82
bsm-week3	**0.97**	**0.97**	0.80	0.92	0.90	0.82	0.78	0.88
snd-cert	0.95	0.96	0.94	**1.00**	**1.00**	0.96	**1.00**	0.97
snd-unm	0.99	0.99	0.96	**1.00**	**1.00**	0.86	0.98	0.97
	0.93	0.94	0.79	0.91	0.89	0.81	0.82	

Table 5. Results on intrusion detection datasets. Bold numbers denote the best results in each row.

5.3 Statistical analysis

Figure 1 shows box plots of the minimum, median, maximum, and interquartile ranges of the true anomalies among the top scoring sequences across all datasets. Ordering the techniques by their average performance, we find negative selection with r-contiguous patterns to perform overall best. However, the difference to the runners-up r-chunk, FSA-z and kNN is quite small. We performed paired two-sided t tests to determine the statistical significance of these differences. The resulting p-values are shown in Table 6. Using a cutoff value of $p = 0.05$, we would conclude that negative selection with r-contiguous patterns performs significantly better than t-STIDE and clustering, while there is no significant difference to kNN, FSA, FSA-z, and to r-chunk pattern. Similarly, we would conclude that negative selection with r-chunk patterns performs significantly better than t-STIDE, clustering, and FSA, while there is no significant difference to FSA-z and kNN.

	fsa	fsaz	stide	knn	cls	rchunk
rcont	0.067	0.510	**0.012**	**0.026**	0.357	0.363
rchunk	**0.046**	0.502	**0.007**	**0.030**	0.363	

Table 6. Resulting p-values of a two-sided t test of the null hypothesis that the corresponding pair of techniques give equal results on our 14 datasets. Those p-values that lie below a cutoff of 0.05 are highlighted in bold.

6 Discussion

To interpret our numerical results for the sliding window based techniques, it is instructive to adopt a characterization by Chandola et al. [13] according to which we can distinguish three types of substrings given a training set S, namely *unseen*, *seen-rare*, and *seen-frequent* substrings. First, note that negative selection algorithms are based only on unseen strings and do not distinguish at all between rare and frequently seen strings. In contrast, the t-STIDE, FSA, and FSA-z techniques do all assess the frequency of seen strings. Therefore, they use in principle more information than negative selection, except FSA which ignores unseen strings. That negative selection still performs competitively can be explained by noting that unseen sequences carry important information, a fact noted already by Chandola et al [13]. This is especially true for the language data, which has the smallest training sets. In this case, there is least information to be found in the frequency distribution of small substrings, explaining why negative selection performs best here. The short length of the input strings is also a problem for the kernel based techniques, which rely on pairwise comparison of the sequences, and therefore gain more information when longer sequences are compared. Thus, they perform very poorly on the language datasets but much better on the other datasets. This link between size of input data and the performance of the different classifiers would be interesting to explore in future work.

In summary, this paper presents what is, to our knowledge, the most comprehensive comparative evaluation of string-based negative selection algorithms on sequence data so far, and the first to use sizable non-binary alphabets. Overall, we found negative selection algorithms to perform competitively to state-of-the-art algorithms, and thus our results contradict earlier studies on artificially generated datasets where negative selection was found to perform very poorly compared to standard statistical techniques [9]. We believe this discrepancy to be mainly explained by the fact that the datasets used by Stibor [9] were generated by first sampling points from a connected shape in two-dimensional space, and then encoding those points as binary strings. In other words, the semantical nature of the data was "hidden" from the r-chunk and r-contiguous patterns, which were therefore not able to generalize properly. Because using r-chunk or r-contiguous patterns brings the assumption that close positions in the input data are more correlated than distant positions, we believe it to be more meaningful to evaluate these algorithms on data where this assumption actually holds, or where at least there is no good reason to assume that this assumption is violated. For the same reason, we agree with Stibor [8] that it is probably not a good idea to use r-chunk or r-contiguous patterns directly on IP packets.

Obviously, our analysis still represents just a starting point, and could be extended in several interesting ways. Our hope is that other members of the community will join us in our quest for a deeper understanding of the strengths and weaknesses of string-based negative selection, using the new tools that are now available. However, it must be said that the algorithms proposed in Elberfeld and Textor [10] and Liśkiewicz and Textor [17] are rather intricate, and implementing

these algorithms was a significant part of the work that underlies this paper. The implementations are available as open source software and can be downloaded from the author's web page at www-binf.bio.uu.nl/textor/negativeselection.html.

References

1. Forrest, S., Perelson, A.S., Allen, L., Cherukuri, R.: Self-nonself discrimination in a computer. In: Proceedings of the IEEE Symposium on Research in Security and Privacy, IEEE Computer Society Press (1994) 202–212
2. Forrest, S., Hofmeyr, S.A., Somayaji, A.: Computer immunology. Communications of the ACM **40** (1997) 88–96
3. Kim, J., Bentley, P.J.: An evaluation of negative selection in an artificial immune system for network intrusion detection. In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO), Morgan Kaufmann (2001) 1330–1337
4. D'haeseleer, P., Forrest, S., Helman, P.: An immunological approach to change detection: Algorithms, analysis, and implications. In: Proceedings of the IEEE Symposium on Security and Privacy, IEEE Computer Society (1996) 110–119
5. Timmis, J., Hone, A., Stibor, T., Clark, E.: Theoretical advances in artificial immune systems. Theoretical Computer Science **403** (2008) 11–32
6. Hofmeyr, S.A., Forrest, S., Somayaji, A.: Intrusion detection using sequences of system calls. Journal of Computer Security **6** (1998) 151–180
7. Balthrop, J., Esponda, F., Forrest, S., Glickman, M.R.: Coverage and generalization in an artificial immune system. In: Proceedings of the 2002 Genetic and Evolutionary Computation Conference (GECCO 2002). (2002) 1045–1050
8. Stibor, T.: On the Appropriateness of Negative Selection for Anomaly Detection and Network Intrusion Detection. PhD thesis, Technische Universität Darmstadt (2006)
9. Stibor, T.: An empirical study of self/non-self discrimination in binary data with a kernel estimator. In: Proceedings of the 7th International Conference on Artificial Immune Systems (ICARIS), Springer (2008) 352–363
10. Elberfeld, M., Textor, J.: Efficient algorithms for string-based negative selection. In: Proceedings of the 8th International Conference on Artificial Immune Systems (ICARIS 2009). Volume 5666 of Lecture Notes in Computer Science., Springer (2009) 109–121
11. Elberfeld, M., Textor, J.: Negative selection algorithms on strings with efficient training and linear-time classification. Theoretical Computer Science **412** (2011) 534–542
12. Stibor, T., Timmis, J., Eckert, C.: The link between r-contiguous detectors and k-CNF satisfiability. In: Proceedings of the Congress on Evolutionary Computation (CEC), IEEE Press (2006) 491–498
13. Chandola, V., Mithal, V., Kumar, V.: A comparative evaluation of anomaly detection techniques for sequence data. In: Proceedings of the 2008 Eighth IEEE International Conference on Data Mining (ICDM '08). (2008) 743–748
14. Moya, M.M., Hush, D.R.: Network constraints and multi-objective optimization for one-class classification. Neural Networks **9**(3) (1996) 463 – 474
15. Chandola, V., Banerjee, A., Kumar, V.: Anomaly detection: A survey. ACM Computing Surveys **41**(3) (2009) 1–58
16. Chapelle, O., Schölkopf, B., Zien, A., eds.: Semi-Supervised Learning (Adaptive Computation and Machine Learning series). The MIT Press (2006)

17. Liśkiewicz, M., Textor, J.: Negative selection algorithms without generating detectors. In: Proceedings of Genetic and Evolutionary Computation Conference (GECCO'10), ACM (2010) 1047–1054
18. Warrender, C., Forrest, S., Pearlmutter, B.: Detecting intrusions using system calls: Alternative data models. In: IEEE Symposium on Security and Privacy, IEEE Computer Society (1999) 133–145
19. Michael, C.C., Ghosh, A.: Two state-based approaches to program-based anomaly detection. In: Proceedings of the 16th Annual Computer Security Applications Conference. (2000) 21
20. Jain, A.K., Dubes, R.C.: Algorithms for Clustering Data. Prentice Hall (1988)
21. Melville, H.: Moby-Dick, or, The Whale. Hendricks House, New York (1952)
22. Bateman, A., Birney, E., Durbin, R., Eddy, S.R., Howe, K.L., Sonnhammer, E.L.: The PFAM protein families database. Nucleic Acids Research **28** (2000) 263–266
23. Forrest, S., Hofmeyr, S.A., Somayaji, A., Longstaff, T.A.: A sense of self for unix processes. In: Proceedings of the IEEE Symposium on Security and Privacy, Washington, DC, USA, IEEE Computer Society (1996) 120–128
24. Lippmann, R.P., et al.: Evaluating intrusion detection systems – the 1998 DARPA offline intrusion detection evaluation. In: DARPA Information Survivability Conference and Exposition (DISCEX) 2000. Volume 2., IEEE Computer Society Press (2000) 12–26