Pöllä M, Honkela T. Negative selection of written language using character multiset statistics. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY 25(6): 1256–1266 Nov. 2010. DOI 10.1007/s11390-010-1099-y

Negative Selection of Written Language Using Character Multiset Statistics

Matti Pöllä and Timo Honkela

Department of Information and Computer Science, School of Science and Technology, Aalto University, P.O. Box 15400 FI-00076 Aalto, Finland

E-mail: {matti.polla, timo.honkela}@tkk.fi

Received July 15, 2008; revised August 6, 2010.

Abstract We study the combination of symbol frequence analysis and negative selection for anomaly detection of discrete sequences where conventional negative selection algorithms are not practical due to data sparsity. Theoretical analysis on ergodic Markov chains is used to outline the properties of the presented anomaly detection algorithm and to predict the probability of successful detection. Simulations are used to evaluate the detection sensitivity and the resolution of the analysis on both generated artificial data and real-world language data including the English Wikipedia. Simulation results on large reference corpora are used to study the effects of the assumptions made in the theoretical model in comparison to real-world data.

Keywords negative selection, anomaly detection, frequency analysis

1 Introduction

Rapidly increasing amounts of digitally stored information are in the form of discrete sequences such as written language, nucleotide sequences and protein sequences. The vast amount of data requires data mining tools to facilitate the retrieval of relevant information from the large collection of source material. A common data mining problem is how to locate and analyze changes in the sequences in order to detect either intentional modifications or unintentional corruption in the data.

The usual approach in integrity analysis of discrete sequences is to use cryptographic hash functions (such as SHA[1] family of hash functions) to compute fixed-length message digests for each sequence and to compare them with the corresponding checksums of modified sequences. While checksum-based analysis provides accurate and reliable results, it has some limitations. First, no information on the location and the magnitude of the changes is gained from the checksums. Secondly, subsections of the sequence cannot be analyzed independently in parallel as the whole sequence is needed to compute the hash.

Recently, a modified version of the negative selection algorithm (NSA)[2] was used[3] to analyze changes in a collection of Wikipedia articles. The results of this experiment showed that a reasonably coarse-grained analysis of the character frequency of the articles contained sufficient information to detect and locate the edited segments in the articles.

In the following, we present a theoretical analysis on locating modifications in a symbol sequence using a combination of negative selection and character frequency analysis as a preprocessing stage. Also, we analyze how a naïve first-order statistical language model can be used to approximate the detection probability.

This paper is organized as follows. In Section 2, we review the general negative selection principle for anomaly detection and its limitations in processing arbitrary discrete sequences. In Section 3, we present the symbol frequency algorithm for anomaly detection in discrete sequences. In Section 4, we present two Markov chain models for analyzing the theoretical detection probabilities. Section 5 includes results of matching detectors with benchmark corpora. We conclude with discussion and conclusions in Sections 6 and 7.

2 Negative Selection Principle

The negative selection algorithm (NSA) by Forrest *et al.*[2] was introduced as a computational model of biological immune systems' way of producing a protection mechanism against unknown pathogens. The NSA performs a binary classification task by discriminating all patterns into either *self* or *non-self* using a collection

Regular Paper

This work was funded by the Academy of Finland under Grant No. 214144.

of detectors which do not match the data in the *self* class (see Fig.1). Correspondingly, any data that has a match with a detector is classified as *non-self*. This process is analogous to the way a biological immune system produces a diverse repertoire① of T-cell receptors and then eliminates the ones which react to the host organism during the maturation phase in the thymus to avoid auto-immune response[5]. The NSA has since been applied to various anomaly detection applications ranging from computer security[6-7] to tool breakage monitoring[8] and novelty detection in time series[9].

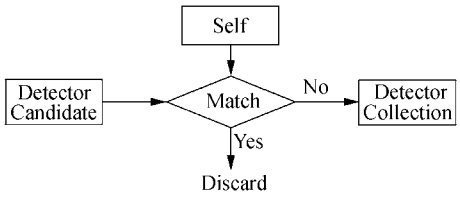

Fig.1. Negative selection of anomaly detectors: a detector candidate is accepted in the detector collection only if it does not match the collection of self samples.

The following properties of NSA have motivated its use as a tool for anomaly detection[10].

Distributability. The overall detection task can be divided into any number of nodes which can operate independently of each other as virtually no communication between the nodes is needed.

Locality. Any subset of the analyzed data can be checked independently for anomalies without the whole data collection being present.

Security. It is very difficult to reverse the information of the detectors to restore the original data (compared to a brute force attack on message digests).

Robustness. The overall anomaly detection system can be made tolerant to some nodes being offline by allowing overlapping targets for the nodes.

Potential disadvantages of negative selection include:

• usually, there are no guarantees on the quality of the system's performance due to the probabilistic nature of the detection process;

• the detection accuracy is highly dependent on the chosen data representation scheme and matching rule[10];

• the required size of the negative description (detector population) can grow prohibitively large making the algorithm impractical in terms of scalability[11].

A formal analysis on the relationship of negative and positive selection has been presented by Esponda *et al.* in [12].

The generic form of NSA is often paired with a constant-length string matching metric to quantify the similarity between two strings. The Levenshtein distance (edit distance) measures string similarities by computing the minimum amount of atomic editing operations (remove/add/substitute) that is needed to transform one string into another[13]. The Hamming distance is a special case of the Levenshtein distance where only substitution operations are considered. A commonly used string metric in AIS literature is the r-contiguous bit rule[14] which computes the longest sequence of identical symbols in two strings. The r-chunk matching rule[15] is a variant of the r-contiguous matching rule which matches two strings of equal length according to the length of matching substrings inside them.

Matching rules which are based on bit-wise similarity metrics become less useful when the size of the symbol vocabulary is larger[16-17]. In particular, it has been shown[11] that the number of evaluations for generating r-contiguous detector grows exponentially for r. Written language is an example of such case where the size of the symbol vocabulary ranges from tens to thousands and the usual string metrics used in NSA applications are not practical. In the following, the symbol frequency negative selection algorithm (SF-NSA) is presented as an alternative to traditional string metrics in analyzing natural language.

3 Strings as Symbol Multisets

3.1 First-Order Statistics of Character Multisets

Although the generic NSA can be directly applied to any binary data[2], the effectiveness of the algorithm is very sensitive to the way in which the similarity metric of data items is defined and the type of data which is being processed as noted in a review of NSA algorithms by Ji and Dasgupta[10]. Especially, the matching rule should be selected such that an effective algorithm can be used in generating the detectors in order to avoid random search[18-19]. Stibor *et al.*[11] have also found that the commonly used r-chunk matching rule is appropriate only in a limited set of problems where the examined data strings are reasonably short (e.g., less than 32 characters). This limitation is problematic since we are interested in analyzing variable-length strings from a large symbol vocabulary.

Domain-specific customizations to the basic NSA involve defining (i) a suitable data representation scheme and (ii) a similarity metric for data items[10]. In the following, the symbol frequence NSA (SF-NSA) algorithm is presented as a modification of the basic NSA by considering the properties of language data.

①Theoretical estimates about the number of unique T-cell receptors (TCRs) are based on the estimated 10^{12} T-cells in the human body[4].

The goal is to produce a collection of detectors which is (a) as compact in size as possible, (b) as likely as possible to match any given sample of text, and (c) does not match the original string S for which the detectors are generated. Using a first-order statistical model for character occurrence, we could thus consider the sequence "e" as the first candidate as it is the most common single-character string in English. However, instead of matching two strings directly, we remove the significance of the character order locally and focus merely on the frequence of each character individually in a subset of the sequence. This effectively means transforming strings into *multisets* (bags) of characters and performing NSA on the character frequencies.

This transformation is motivated by the variance of the character frequency distribution found in natural language. In Fig.2(a) the character frequency distribution of the Reuters corpus[20] is shown. In Fig.2(b), the Kullback-Leibler distance from this overall distribution is shown for different subsets of the corpus. There is much variance in the distributions even for longer subsets of the corpus which indicates that it is possible to find features in the symbol frequence profile which are common in general, but absent in S.

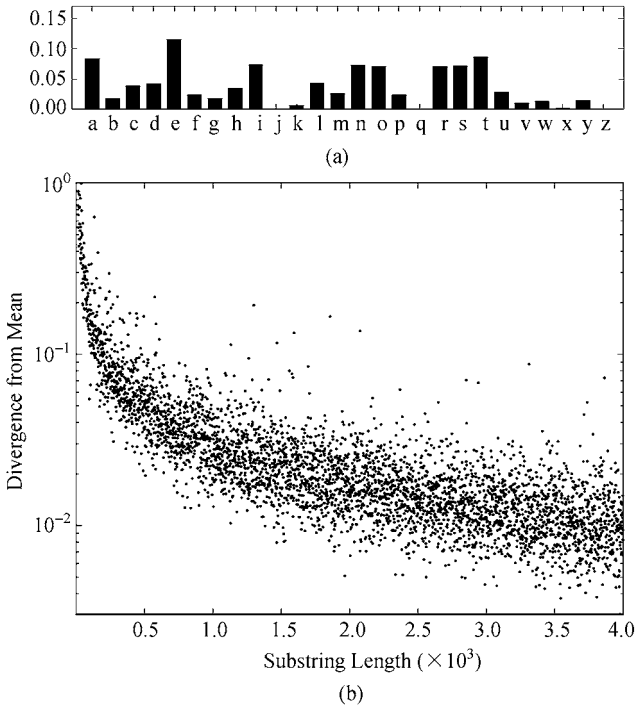

Fig.2. Variance in the character frequence distributions. Kullback-Leibler divergence from the mean distribution is shown for random samples of 1 to 4 000 characters from the Reuters corpus.

3.2 Data Representation

The SF-NSA algorithm is based on a transformation where an arbitrary-length string S from a symbol vocabulary Σ is mapped into a binary matrix which describes the frequency of each character in Σ in a window of w adjacent characters.

In the transformation, the frequency $0 \leqslant f_c \leqslant w$ of each character $c \in \Sigma$ is computed in each position of a sliding window of w characters. After this process, the frequencies f_c which were not observed in string S contain a negative description about S.

The detection of an anomaly is based on the probability of the anomaly changing this negative description. The detector is defined as follows.

Definition 1. *An SF-NSA detector (c, w, k) is defined by a character c, a window length w and a target frequency k for c.*

3.3 Matching Rule

A match between a detector (c, w, k) and a string S occurs if the target frequency for the specified character is observed in the string.

Definition 2. *A string S has a match with a detector (c, w, k) if there is a position in the string where the frequency of character c in the set of w adjacent characters is k.*

3.4 Example

As a minimal example case, consider the sequence "abacb" using a window length $w = 3$. There are three positions for the sliding window in which the frequence of 'a' is computed as follows.

a	b	a	c	b	f_a	f_b	f_c
a	b	a			2	1	0
	b	a	c		1	1	1
		a	c	b	1	1	1

The frequency f_a has values 1 and 2, but not 0 or 3. By repeating the same procedure for 'b' and 'c', we end up with the following detectors:

('a', 3, 0) ('a', 3, 3) ('b', 3, 0) ('b', 3, 2)
('b', 3, 3) ('c', 3, 2) ('c', 3, 3).

A random edit to the original string is likely to change the character statistics such that the above restrictions for the character frequencies will not be satisfied and the change can be detected — even without knowing the contents of the original string. For instance, consider a case where the string is changed into "ababcb". When comparing the changed string with the detector collection the following matches are found:

(a, 3, 0) aba**bcb**
(b, 3, 2) a**bab**cb
(b, 3, 2) aba**bcb**.

In the following, the properties of this detection process is analyzed for the general case of any string and window length.

4 Theoretical Analysis of SF-NSA

4.1 String/Multiset Transformation

To limit the required size of the negative description in NSA algorithms, a single detector should be able to match several items in the non-self space. Usually, this is implemented as a threshold in a bitwise matching rule or a spherical neighborhood in real-valued NSA[21]. In SF-NSA, the one-to-many matching is enabled by the transformation from strings into character multisets (bags) which all have the same character frequency statistics.

The number of unique multisets is given by the multiset coefficient

$$\left(\!\!\binom{|\Sigma|}{w}\!\!\right) = \binom{|\Sigma| + w - 1}{w} \tag{1}$$

where w is the number of characters in the multiset and $|\Sigma|$ is the size of the symbol vocabulary.

This phenomenon is illustrated in Fig.3(a) where the exponentially growing number of unique combinations of 26 characters ('a' to 'z') is shown for various string lengths with the corresponding number of unique multisets. Also, the corresponding observed numbers in the Reuters corpus are shown. The theoretical and observed string/multiset ratios are shown in Fig.3(b). It is worthwhile to note that while the theoretical strings-per-multiset ratio increases with w, the maximum ratio in the observed data in the Reuters corpus is at its maximum at $w = 5$. This is due to the sparsity of a random w-length string being valid language. (See Table 1).

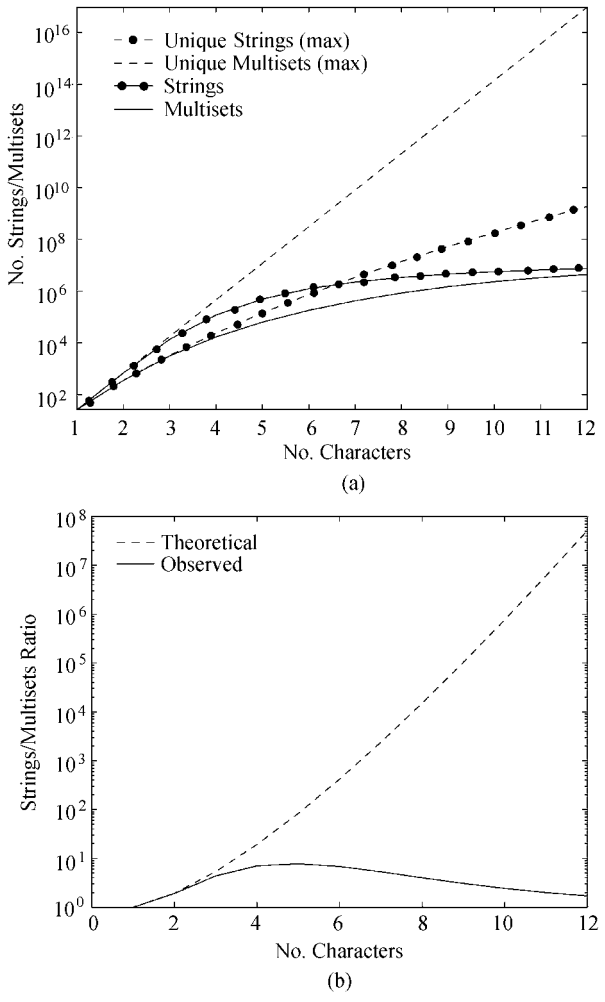

Fig.3. Theoretical upper limits for the amount of unique character combinations and character multisets and the observed quantities in a sample from the Reuters corpus.

Table 1. Number of Observed Unique Character Multisets and Strings in the Reuters Corpus for Window Lengths 1 to 12

w	Unique Multisets	Unique Strings	Ratio
1	26	26	1.00
2	351	672	1.91
3	3 080	13 491	4.38
4	16 820	118 470	7.04
5	62 984	488 588	**7.76**
6	179 594	1 227 358	6.83
7	420 893	2 237 896	5.32
8	841 540	3 373 975	4.01
9	1 472 950	4 515 598	3.07
10	2 307 255	5 595 144	2.43
11	3 293 964	6 575 533	2.00
12	4 356 104	7 431 806	1.70

When considering a single detector (c, w, k), there are $\binom{w}{k}$ possible positions for the character c to appear in a matching string. Also, considering that the remaining $(w - k)$ characters have no significance, there are a total of

$$\binom{w}{k}(|\Sigma| - 1)^{w-k} \tag{2}$$

unique strings that match a single detector.

In the following, we analyze the probability of a given detector matching a given sequence.

4.2 Detector Match Probability

The Bernoulli trial can be used to analyze the probability of a given detector matching a random text segment. We thus assume that text is a sequence of characters which appear independently of each other according to the prior probabilities of each character. The probability of observing exactly k occurrences of character c in a sequence of w adjacent characters is given

by the Binomial distribution

$$P(K=k)=\binom{w}{k}p^k(1-p)^{w-k} \tag{3}$$

where p is the prior probability of observing character c. This gives the match probability for a text segment which is only as long as the window. For analyzing the character statistics of each position of the sliding window, we can apply a Markov model.

4.3 Markov Model 1

When matching a given detector (c,w,k) to a string of l_S characters there are l_S-w+1 positions i for the window. The match probability at the initial step is given by (3). The process of sliding the window across the string can be viewed as an ergodic and regular Markov process (a process in which every state can be reached from every state in some number of steps[22]). Each state of the process is defined by a w-length bit string $\{b_1,b_2,\ldots,b_w\}$ in which b_n is 1 if the character at position n is c and otherwise 0. The starting probability distribution $\boldsymbol{u}$ of the chain is

$$u_i=p^k(1-p)^{w-k} \tag{4}$$

where k is the sum of the bits b_n of state i (see Fig.4).

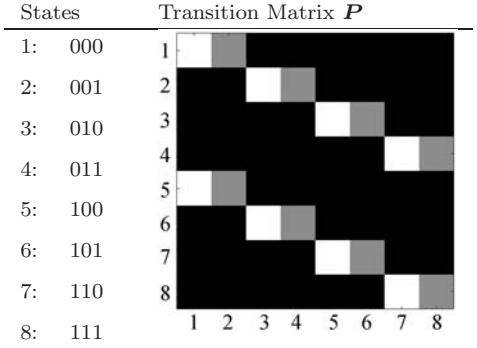

Fig.4. States and transition probability matrix for $w=3$ and $p=0.4$. In the matrix light shades correspond to large values of p_{ij}.

There are thus 2^w states for which the transition probabilities can be defined using a $2^w\times 2^w$ transition matrix

$$\boldsymbol{P}=\begin{bmatrix} p_{1,1} & \cdots & p_{2^w,1} \\ \vdots & \ddots & \vdots \\ p_{1,2^w} & \cdots & p_{2^w,2^w} \end{bmatrix}. \tag{5}$$

At each step the bit positions are shifted left such that the leftmost bit b_1 is removed and a new bit enters the rightmost position b_w. Using the first-order model, the probability of the next character being c is p. This allows us to define the transition probabilities for each transition in (5) as follows

$$p_{ij}=\begin{cases} p, & \text{if } b_n^{i-1}=b_{n+1}^j\ \forall n\in[1,w-1],\ b_w^j=1,\\ 1-p, & \text{if } b_n^{i-1}=b_{n+1}^j\ \forall n\in[1,w-1],\ b_w^j=0,\\ 0, & \text{otherwise,}\end{cases} \tag{6}$$

where the first case corresponds to the next character being c, the second case to any other character.

The match probability now has an interpretation using the Markov chain: the probability of a detector (c,w,k) matching a random string of length l_S is equal to the probability of the Markov process visiting a state where $\sum_{i=1}^w b_i=k$ at least once in l_S-w+1 steps. This probability can be expressed using the complementary event — the event that the process does not visit any such state.

$$P_M\big((c,w,k),l_S\big)=1-\left(\boldsymbol{u}\prod_{t=0}^{l_S-w+1}\left(1-\sum_s \boldsymbol{P}^t(i,s_k)\right)\right) \tag{7}$$

where $\boldsymbol{u}$ is a vector representing the starting probability distribution and $\boldsymbol{P}^t(i,k)$ is the transition probability from state i to state k in exactly t steps and s_k is any state for which $\sum_{i=1}^w b_i=k$.

Analysis using this model becomes difficult for larger window lengths as the size of the transition matrix grows in $\mathcal{O}(w^2)$. Using the equal probabilities of all states for which $\sum_{n=1}^w b_n=k$ we can use a simplified Markov model.

4.4 Markov Model 2

To reduce the state space of the first model, we can consider the group of states for which $\sum_{n=1}^w b_n=k$. For example the states $001,010$ and 100 for $w=3$ and $k=3$ in the example (see Fig.6). Each of these states has the same prior probability defined by (4). At each step the process can either move to

(i) a state where $\sum_{n=1}^w b_n=k+1$,

(ii) a state where $\sum_{n=1}^w b_n=k-1$,

(iii) some other state in which $\sum_{n=1}^w b_n=k$.

The probability of these three mutually exclusive events is defined by the new incoming character and the leftmost bit in the current state. When all of these $\binom{w}{k}$ states are combined into a single state we have a new ergodic Markov chain with $w+1$ states describing the random variable k in the set of w adjacent characters with a transition matrix defined by

$$\boldsymbol{P}=\begin{bmatrix} p_{00} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & p_{ww} \end{bmatrix} \tag{8}$$

where the transition probabilities p_{ij} are

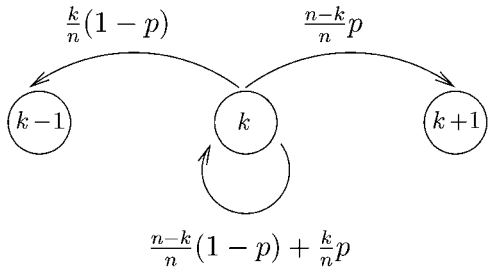

Fig.5. Frequence of character c in a window of w characters described as an ergodic state Markov chain.

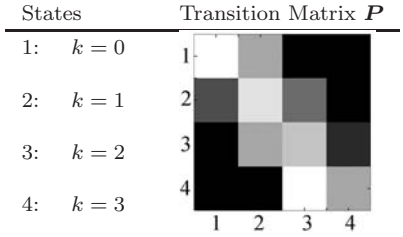

Fig.6. States and transition probability matrix for $w = 3$ and $p = 0.4$. In the matrix light shades correspond to large values of p_{ij}.

$$p_{ij} = \begin{cases} \dfrac{n-k}{n}(1-p), & j = i+1, \\ \dfrac{k}{n}p, & j = i-1, \\ \dfrac{n-k}{n}(1-p) + \dfrac{k}{n}p, & j = i, \\ 0, & \text{otherwise.} \end{cases} \quad (9)$$

Using (3) the stationary distribution $\boldsymbol{u}$ for the $w+1$ states is

$$u_i = \binom{w}{u_i} p^{u_i}(1-p)^{w-u_i}.$$

The reduced form of the state space and the transition matrix of Fig.4 is shown in Fig.6.

Again, we can analyze the match probability as the probability of this process visiting state k at least once in $l_S - w + 1$ steps corresponding to all possible positions of the sliding window. For this we use the complement event of the process going through $l_S - w + 1$ steps without visiting state k.

$$P_M\big((c,w,k), l_S\big) = 1 - \Big(\boldsymbol{u} \prod_{t=0}^{l_S-w+1} (1 - \boldsymbol{P}^t(i,k))\Big) \quad (10)$$

which is the match probability of a detector (c, w, k) for a random string of length l_S.

4.5 Computational Complexity

The original motivation for using SF-NSA is the limitation of standard string similarity metrics in a universe of $|\Sigma|^w$ possible strings. Especially, using random search in the process of generating detectors would not be practical. Using SF-NSA, the size of the initial detector repertoire is $|\Sigma|(w+1)$.

Even the total amount for required character histogram computations grows linearly with the size of the analyzed document as the total amount of histogram comparisons is $|\Sigma|(w+1)(D-w+1)$.

5 Simulations

In the following experiment settings, we evaluate the performance of SF-NSA in detecting a given change in a string and the probability of a given detector of matching a randomly selected segment of natural language. These simulations are presented to supplement the earlier work[3] on the performance of SF-NSA with real-life data.

5.1 Detection Rate

The properties of the matching process guarantee that if no change has occurred in the analyzed string S, then the test result is also negative. Hence, the probability of false positives (FP) is zero since the negative selection has already eliminated all detectors which match S. For this reason, the ROC characteristics (tradeoff between true positives and false positives) are not descriptive and the interest is mainly in the sensitivity of the detection.

In the following section, the detection rate (sensitivity) is used as a measure of the efficiency of the detection process. The detection rate d is defined as the ratio of true positives (TP) to the total amount of modified strings (true positives and false negatives (FN)).

$$\text{detection rate} = \frac{TP}{TP + FN}. \quad (11)$$

The detection rate is compared with the selected value of w since there is a tradeoff between sensitivity and the compactness of the detector collection: high accuracy will require a high value of w which leads to a larger collection of non-self detectors.

5.2 Random Modification of Individual Characters

An experiment of changing random characters in random positions of a string was conducted to study the sensitivity of the detection for short strings. An average result computed from 1000 trials was recorded for modifying a single character in a randomly generated string of 20, 40 and 60 characters from a vocabulary of 26 characters with a uniform probability distribution. The same experiment was then repeated for three and ten changed characters. The window length of the detectors was varied from 1 to 18 characters.

The results are shown in Fig.7 and Table 2. As seen in the figures, the window length parameter has a significant effect on the performance. In the case of detecting a change of a single character in a string of 60 characters, the accuracy grows from 10% to 55% when the window size is increased from 1 to 18. A notable result is the case of detecting 10 character modifications in a string of 60 characters: a small window of 4 characters is enough to detect the anomaly with 90% accuracy.

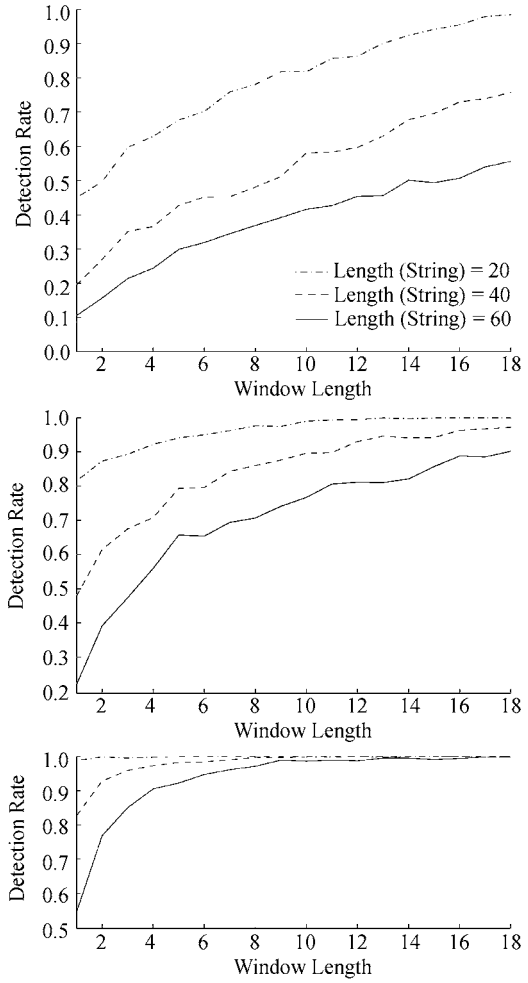

Fig.7. Detection rate of randomly modifying 1 (a), 3 (b) and 10 (c) characters in a string of 20 (dash-dot line), 40 (dashed line) and 60 (solid line) characters.

Table 2. Detection Rates for Identifying Randomly Shifted Characters Inside a Symbol String (Results for modifying 1/3/10 characters for strings of length 20/40/60)

w	Δ1 Char			Δ3 Chars			Δ10 Chars		
	20	40	60	20	40	60	20	40	60
1	0.45	0.19	0.10	0.81	0.47	0.22	0.98	0.82	0.54
2	0.49	0.27	0.15	0.87	0.61	0.39	0.99	0.92	0.76
4	0.62	0.36	0.24	0.92	0.70	0.56	0.99	0.97	0.90
6	0.70	0.45	0.31	0.95	0.79	0.65	1.00	0.98	0.94
8	0.78	0.48	0.36	0.97	0.86	0.70	1.00	0.99	0.97
10	0.81	0.58	0.41	0.99	0.89	0.76	1.00	0.99	0.98
12	0.86	0.59	0.45	0.99	0.93	0.81	1.00	1.00	0.98
14	0.92	0.67	0.50	0.99	0.94	0.82	1.00	0.99	0.99
16	0.95	0.73	0.50	1.00	0.96	0.88	1.00	1.00	0.99
18	0.98	0.75	0.55	0.99	0.97	0.90	1.00	1.00	0.99

5.3 Detecting Modification in Wikipedia Articles

As an example of using the SF-NSA algorithm in a practical setting we use a corpus of randomly selected Wikipedia articles. For each of the 500 articles, there exist two versions and the SF-NSA is used to detect the location of the edit. These edits range from small corrections to details to complete rewrites of the articles. As a preprocessing phase, we remove all meta-information from the articles. Also, punctuation characters are removed and the analysis ignores text case to limit the symbol vocabulary to 27 characters (a∼z and a whitespace symbol).

We use the SF-NSA for window sizes 2∼170 to detect the edit between two successive versions of an article. The initial detector pool of $(w+1)|\Sigma|$ detectors is pruned such that all matches with the first version of the article are removed. Any match between the remaining set of detectors with the other article version is classified as a modification.

For comparison, we computed a baseline result using the symbol frequency statics of w all adjacent characters instead of analyzing the frequencies individually for each character. Specifically, we compared the resulting character histogram sets of both article versions to see whether some character multisets are unique to one of the two versions. This result also gives an useful upper limit in character frequency-based anomaly detection as the result for per-character analysis can at best only be as good as the baseline result.

The results of the Wikipedia experiment are presented in Table 3 and Fig.8. The detection rate of

Table 3. Detection Rates for the Wikipedia Edit Detection Experiment

w	Baseline Detection Rate	SF-NSA Detection Rate
2	0.43	0.12
5	0.90	0.21
8	0.94	0.27
15	0.97	0.35
30	0.98	0.45
50	0.98	0.55
70	0.98	0.63
90	0.98	0.69
110	0.98	0.72
130	0.98	0.75
150	0.98	0.78
170	0.98	0.80

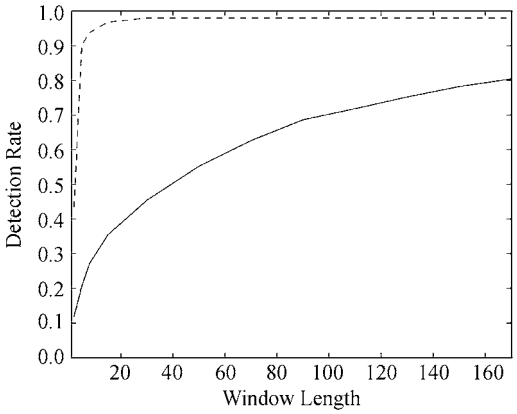

Fig.8. Detection rate for analyzing changes in Wikipedia articles ($N = 500$) using various window lengths. For comparison, the dashed line gives a baseline result of the best possible detection using symbol-frequencies.

SF-NSA exceeds 0.5 for window lengths $w > 40$ and reaches 0.8 for $w = 170$. As expected, the baseline result of analyzing all characters of the multiset is significantly better and the symbol frequency statistics of only five adjacent characters is enough to detect 90% of the modifications.

5.4 Detector Sensitivities Using Large Corpora

To compare the theoretical results of Section 4, we used a group of well-known benchmark corpora to measure the matches rates of SF-NSA detectors (i.e., the proportion of random samples from the corpus that match the detector). For various detectors (c, w, k) we repeated a test where a random position in the corpus was selected and the matching was done for m steps (simulating the string length) to study the match probability. A preprocessing stage of converting the strings into lower case and removal of whitespace and punctuation was applied to normalization. The used corpora are as follows.

Reuters. A large collection of Reuters news articles[20] (black dots).

Europarl. A corpus extracted from the proceedings of the European Parliament[23] (blue dots).

Ulysses. Novel by James Joyce, downloaded from Project Gutenberg② (green dots).

Grimm. The *Household Stories by the Brothers Grimm by Jacob Grimm and Wilhelm Grimm*, downloaded from Project Gutenberg (magenta dots).

Kalevala. The national epic of Finland downloaded from Project Gutenberg (cyan dots).

Random. A corpus generated randomly using the first-order character statistics of the Reuters corpus (black circles).

In Fig.9 the match rates (proportion of found matches in the total number of trials) for detectors ($c =$ 'e', $w = 2$, $f_c = 0..2$) are presented. As the most common character in English (with $p = 0.1147$ in the Reuters corpus) these are one of the most interesting ones. The interpretation for the sensitivity of ('e', 2, 0) is straightforward: it is easy to find two adjacent characters where 'e' does not appear. This property is present in all of the corpora as well as the theoretical result (dashed line). For ('e', 2, 0) we see that about 90% of strings of length 20 contain one 'e'. The first deviation from the theoretical model is seen in the match rate of ('e', 2, 2) (Fig.9(c)). The theoretical model predicts a higher match rate for this detector.

In Fig.10 the match rates for a detector with longer window length is shown. Looking at the varying target frequencies from 0 to 7 we see that the steepest increase in the match rate is found in ('a', 30, 4) and ('a', 30, 5). For the Kalevala corpus the result is different due to the different prior frequency of 'a' in the corpora (0.0846 in Reuters, 0.1174 in Kalevala).

The inaccuracy caused by the first-order language model which does not consider the joint probabilities of

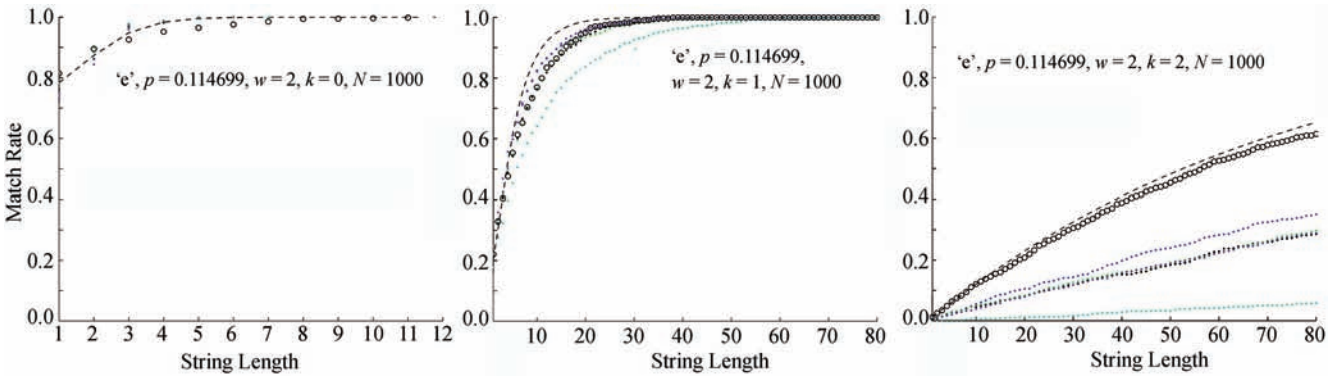

Fig.9. Match rates (proportion of matching random samples) for ('e', 2, 0..2).

②http://www.gutenberg.org/.

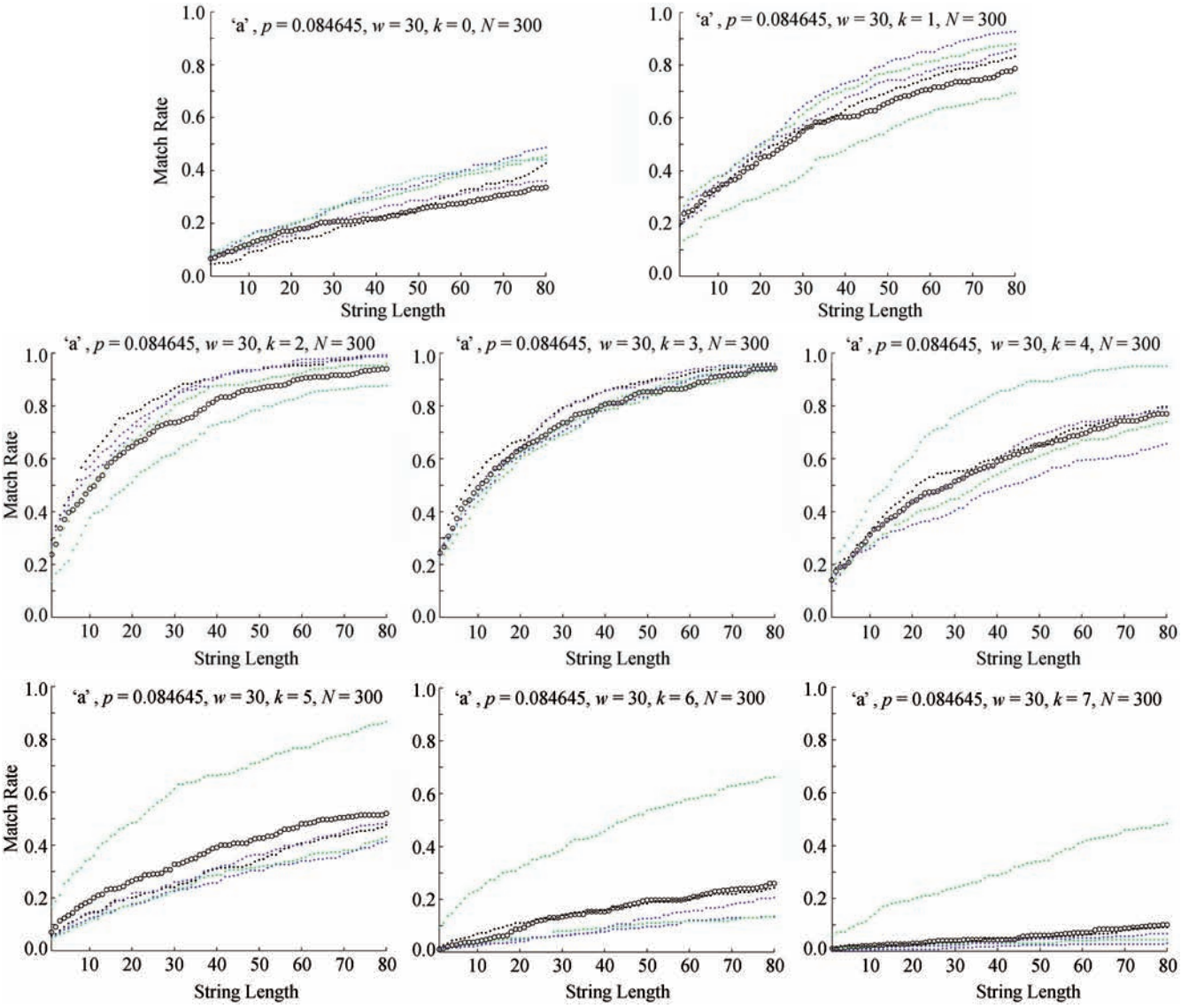

Fig.10. Match rates (proportion of matching random samples) for detectors ('a', 30, 0..7).

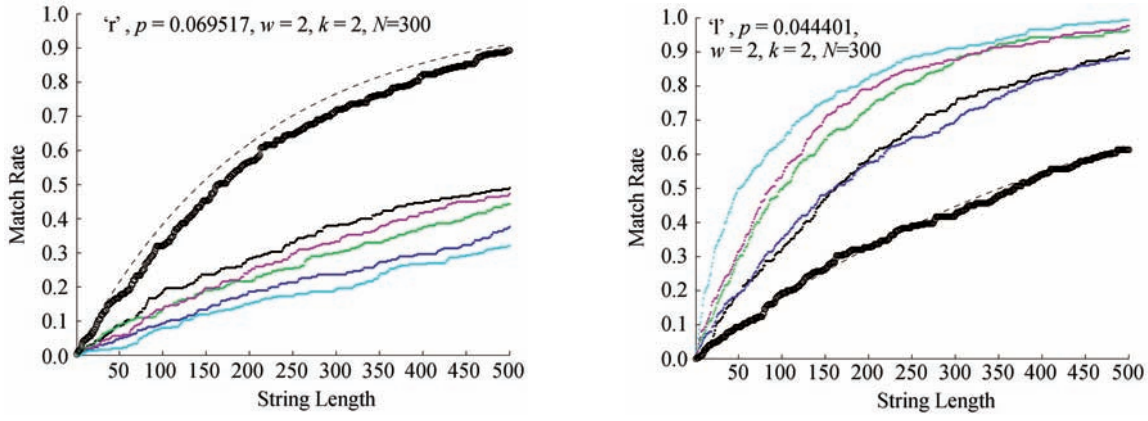

Fig.11. Inaccuracies of the first-order model: 'rr' is less common and 'll' more common than the first-order model would predict.

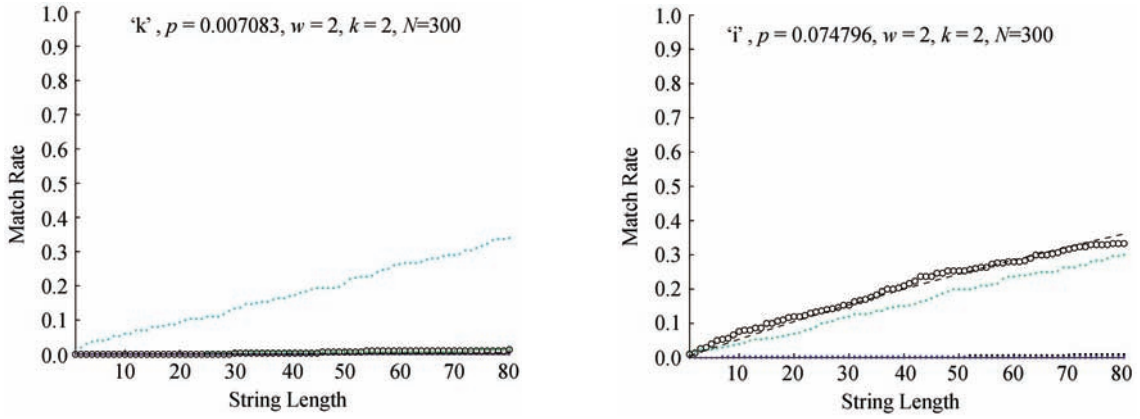

Fig.12. Match rates (proportion of matching random samples) for ('k', 2, 2) and ('i', 2, 2) illustrating the sparsity of the pattern 'kk' (properly represented by the first-order model) and the sparsity of 'ii' (not captured by the first-order model). The Kalevala corpus stands out in this result due to the occurrence of gemination in Finnish.

two characters can be seen in Fig.11(a). The theoretical result (dashed line) predicts the segment 'rr' being much more common than what is observed in the corpora. Another result is shown in Fig.11(b), where the match rate of ('l', 2, 2) shows that the segment 'll' is by far more common in used language than the first-order model would predict. The difference between the English corpora and the Kalevala corpus is seen in Fig.12 where the match rates for ('k',2, 2) and ('i', 2, 2) are shown.

6 Discussion and Future Work

We have presented a theoretical framework based on ergodic Markov chains to analyze the detection probability of SF-NSA: a negative selection algorithm for change detection of large-vocabulary strings of arbitrary length. As a follow-up to previous experiment on real-life data, we find the presented theoretical work a valuable addition to the recent trend towards a theoretical approach in the research of AIS[24].

As demonstrated in the simulations of Section 5 the detection result for small changes is highly dependent on the size of the original document, the used window length and the magnitude of the changes. Specifically, a window length of merely 4 characters was enough to gain a 90% sensitivity for detecting a change of 10 characters in a string of $l_S = 60$.

Throughout the analysis, SF-NSA has been presented in a way which is independent of the used language or writing system. Although the theoretical results of Section 4 and the simulations of Section 5 use only the 26 letters of the English alphabet ('a' to 'z') the algorithm can be directly applied to any set of characters in the Unicode standard which covers virtually all known writing systems with potentially over a million code points[25]. It is expected that the presented experimental results would be different in the case of a language with a larger symbol vocabulary. A comparison of the quality of detection for different languages is an interesting future research topic due to the varying (first-order) statistical properties of different languages and the variety in the size of the symbol vocabulary.

Although the SF-NSA trades accuracy for a small detector base, the baseline result of the Wikipedia edit detection experiment is promising in terms of using character frequency information for anomaly detection in strings. Ongoing research on using generative mixture models for anomaly detection in strings has used this approach with promising results[26]. In a recent evaluation paper[27], the SF-NSA is compared to other windowing-based anomaly detection methods including the one-class support vector machine. Outside the domain of natural language, Stibor[28] has applied an n-gram representation in detecting computer viruses in executable binary files.

7 Conclusions

Applying the NSA to sparse data such as written language requires special attention in terms of matching rules and representation of artificial antibodies due to the sparsity of data. This study has introduced a general method based on first-order statistics of character strings to implement a compact negative representation of text documents and for detecting changes in them. The distinctive feature of this method is to apply a special matching rule which compares the statistics of two strings instead of traditional bit-wise matching rules. As a side product this matching rule has the ability to detect deletions, which is a well known pitfall of many NSA algorithms.

We have shown that a reasonably simple Markov model and a prior character probability distribution can be effective in predicting the detection sensitivity of

SF-NSA. Further, this model can be used in optimized versions of the detector generating algorithm.

References

[1] National Institute of Standards and Technology (NIST). FIPS 180-2: Secure Hash Standard, August 2002. Available online at http://itl.nist.gov/fipspubs./

[2] Forrest S, Perelson A S, Allen L, Cherukuri R. Self-nonself discrimination in a computer. In *Proc. the 1994 IEEE Symposium on Research in Security and Privacy*, Oakland, USA, May 16-18, 1994, pp.202-212.

[3] Pöllä M, Honkela T. Change detection of text documents using negative first-order statistics. In *Proc. the Second International and Interdisciplinary Conference on Adaptive Knowledge Representation and Reasoning* (*AKRR 2008*), Porvoo, Finland, Sept. 17-19, 2008, pp.48-55.

[4] Arstila T P, Casrouge A, Baron V, Even J, Kanellopoulos J, Kourilsky P. A direct estimate of the human α β T cell receptor diversity. *Science*, Oct. 1999, 286(29): 958-961.

[5] Leandro N. de Castro, Jonathan Timmis (eds.). Artificial Immune Systems: A New Computational Intelligence Approach. Springer-Verlag, 2002.

[6] Forrest S, Hofmeyr S A, Somayaji A, Longstaff T A. A sense of self for UNIX processes. In *Proc. the 1996 IEEE Symp. Security and Privacy*, Oakland, USA, May 6-8, 1996, pp.120-128.

[7] Hofmeyr S A, Forrest S, Somayaji A. Intrusion detection using sequences of system calls. *Journal of Computer Security*, 1998, 6(3): 151-180.

[8] Dasgupta D, Forrest S. Tool breakage detection in milling operations using a negative-selection algorithm. Technical Report CS95-5, Dept. Computer Science, Univ. New Mexico, 1995.

[9] Dasgupta D, Forrest S. Novelty detection in time series data using ideas from immunology. In *Proc. The International Conference on Intelligent Systems*, 1995.

[10] Ji Z, D Dasgupta. Revisiting negative selection algorithms. *Evolutionary Computation*, 2007, 15(2): 223-251.

[11] Stibor T, Timmis J, Eckert C. The link between r-contiguous detectors and k-CNF satisfiability. In *Proc. Congress on Evolutionary Computation* (*CEC*), Vancouver, Canada, Jul. 2006, pp.491-498.

[12] Esponda F, Forrest S, Helman P. A formal framework for positive and negative detection. *IEEE Transactions on Systems, Man, and Cybernetics*, 2004, 34(1): 357-373.

[13] Fischer I. Pattern recognition algorithms for symbol strings [Ph.D. Dissertation]. University of Tübingen, 2003.

[14] Percus J K, Percus O, Perelson A S. Predicting the size of the antibody combining region from consideration of efficient self/non-self discrimination. *Proc. the National Academy of Science of the USA*, 1993, 90(5): 1691-1695.

[15] Balthrop J, Esponda F, Forrest S, Glickman M. Coverage and generalization in an artificial immune system. In *Proc. GECCO-2002*, New York, USA, July 9-13, 2002, pp.3-10.

[16] Stibor T, Bayarou K M, Eckert C. An investigation of R-chunk detector generation on higher alphabets. In *Proc. GECCO*, Seattle, USA, Jun. 26-30, 2004, pp.299-307.

[17] Stibor T. On the appropriateness of negative selection for anomaly detection and network intrusion detection [Ph.D. Dissertation]. Technische Universität Darmstadt, 2006.

[18] D'haeseleer P, Forrest S, Helman P. An immunological approach to change detection: Algorithms, analysis, and implications. In *Proc. the Symposium on Research in Security and Privacy*, Oaklands, USA, May 6-8, 1996, pp.110-119.

[19] D'haeseleer P. An immunological approach to change detection: Theoretical results. In *Proc. the 9th Computer Security Foundations Workshop*, Dromquinna Manor, Ireland, Mar. 10-12, 1996, pp.18-26.

[20] Lewis D D, Yang Y, Rose T, Li F. RCV1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research*, 2004, 5: 361-397.

[21] González F A, Dasgupta D. Anomaly detection using real-valued negative selection. *Genetic Programming and Evolvable Machines*, 2003, 4(4): 383-403.

[22] Grinstead C M, Snell L J. Introduction to Probability. American Mathematical Society, 4 July, 2006 edition, 2006.

[23] Koehn P. Europarl: A Parallel Corpus for Statistical Machine Translation. MT Summit, 2005.

[24] Timmis J, Hone A, Stibor T, Clark E. Theoretical advances in artificial immune systems. *Theoretical Computer Science*, 2008, 403(1): 11-32.

[25] The Unicode Consortium. The Unicode Standard, Version 5.0. Addison-Wesley Professional, 5th Edition, Nov. 2006.

[26] Pöllä M. A generative model for self/non-self discrimination in strings. In *Proc. Int. Conf. Adaptive and Natural Computing Algorithms*, Kuopio, Finland, Apr. 23-25, 2009, pp.293-302.

[27] Pöllä M. An evaluation of windowing-based anomaly detection schemes for discrete sequences. 2010, unpublished manuscript.

[28] Stibor T. A study of detecting computer viruses in real-infected files in the n-gram representation with machine learning methods. In *Proc. the 23rd International Conference on Industrial, Engineering & Other Applications of Applied Intelligent Systems* (*IEA-AIE*), 2010. (Accepted)

Matti Pöllä is a researcher at Aalto School of Science and Technology, Department of Informationand Computer Science. He received his M.Sc. (Tech.) degree from Helsinki University of Technology (TKK) in 2005 and is currently pursuing the Ph.D. degree in computer science. His research area is biologically inspired computation applied to text mining.

Timo Honkela holds a Ph.D. degree from the Helsinki University of Technology (TKK). He has worked as a professor at TKK and the University of Art and Design Helsinki. He is appointed as chief scientist at the cognitive systems research group of the Adaptive Informatics Research Center at Aalto University School of Science and Technology. His research area is computational cognitive systems.