

CHANGE DETECTION OF TEXT DOCUMENTS USING NEGATIVE FIRST-ORDER STATISTICS

Matti Pöllä and Timo Honkela

Adaptive Informatics Research Centre, Helsinki University of Technology,
P.O.Box 5400, FIN-02015 TKK, FINLAND, firstname.lastname@tkk.fi

ABSTRACT

We present a probabilistic method for change detection in text documents based on the biologically motivated principle of negative selection. Compared to standard checksum-based analysis, our statistical approach is able to locate and approximate the magnitude of changes. Further, the detection process can be distributed to any number of independent nodes resulting in a fault tolerant system. The negative representation of information also makes it possible to protect the privacy of the analyzed data due to the difficulty of reversing the information of the non-self detectors. An experiment with a collection of Wikipedia articles is used to analyze the length of the required negative description compared to the length of the document.

1. INTRODUCTION

Cryptographic hash functions are commonly applied in tasks where the integrity of a message needs to be verified. By comparing fixed-length checksums of two files one can detect any modification in the original message as even the smallest modification in the original file will result in a different checksum. Checksum-based analysis becomes less useful in situations where approximate information on the magnitude and the location of the change inside the file is needed as it can only discriminate whether or not a change has occurred. Locating the change would require computing the hash for every non-overlapping atomic part of the document.

This approach is used in the rsync algorithm [1] which is used to synchronize two versions of a file over a high-latency low-bandwidth link. To reduce the amount of computations, rsync uses a combination of an unreliable but cheap hash algorithm and a reliable but computationally expensive hash function to minimize the amount of required computation. In computer virus detection, Di Crescenzo and Vakil [2] have used an optimized hashing method to locate viruses inside files.

A typical new setting where large amounts of text documents need to be monitored for changes is web site quality auditing. For example, the web has become a popular source for medical advice surpassing the popularity of medical self help books. While the dissemination of medical information in the web has had positive effects, there is a problem related to the difficulty of assessing the quality of medical information online [3, 4]. There is an

increasing need for trusted authorities who can audit the contents of a web site and issue the site a quality label provided that a set of quality criteria are met. But once a web site has been audited, it has to be monitored for changes in case the contents of the site changes significantly. In this case, the quality labeling authority cannot rely on the site administration to provide information on the changes and a manual analysis of the changes would require too much human effort. An automated tool for the tracking of changes on the monitored web sites is thus needed to analyze the contents of the site and alert a human expert in the case of significant changes. Checksum-based monitoring tools, though applicable, are not practical since the large amount of small irrelevant changes would result in a flood of false-positives.

Integrity analysis of web content is relevant also in web traffic analysis. Reis et al. [5] have recently built a system for in-flight change detection for analyzing possible modifications in the process of transferring documents from the hosting server onto the client.

In the following, we present a probabilistic approach for detecting changes in text documents using a modified version of the negative selection algorithm (NSA) [6]. In Section 2 we review the basic NSA algorithm and discuss its applicability to language data. In Section 3 we present Text-NSA which employs a matching rule based on the first-order character statistics of the analyzed strings. In Section 4 we perform an experiment in analyzing changes in a collection of Wikipedia articles and some conclusions are made in Section 5.

2. NEGATIVE REPRESENTATIONS OF INFORMATION

The negative selection algorithm, as originally proposed by Forrest et al. [6], is a computational model of the procedure which biological immune systems of vertebrates use to detect unknown molecules in the body in a robust and distributed manner. The NSA solves a binary classification problem $f : \mathbb{R}^n \rightarrow \{0, 1\}$ of dividing data patterns to either *self* (S) or *non-self* (N) in a shape space U using a population of non-self detectors and a matching rule such that

$$U = S \cup N \text{ and } S \cap N = \emptyset. \quad (1)$$

In contrast to typical classification problems, the NSA assumes that training samples are available only from the self class.

In the initial phase of the NSA a population of detectors is produced as candidates for detecting non-self items. Various domain-specific algorithms can be used to avoid random selection in producing the initial detector candidates [7, 8]. Next, each of the available samples of self data is matched against the detector candidates and all matching detectors are censored out of the detector repertoire (Fig. 1). After the censoring phase, the remaining detectors are matched against new incoming data and any matching items are classified as non-self.

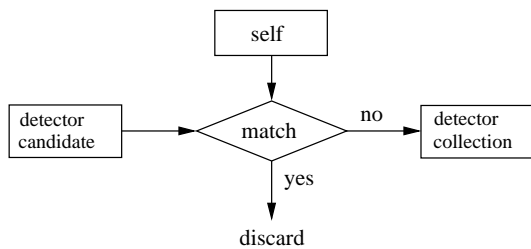


Figure 1. Negative selection in producing a non-self matching detector collection.

This process is considered similar to the way the adaptive biological immune system produces non-self matching lymphocyte cells by removing the ones which respond to the stimulus of the host organism during the maturation phase in the thymus [9]. The principle of negative selection has been adapted to various applications of change detection ranging from computer network security [10] to monitoring of industrial processes [11].

In the following, we present a modified version of the NSA employing first-order statistics (character histograms) of text segments and corresponding version of the NSA matching rule and a non-self detector scheme.

3. TEXT-NSA

Although the generic NSA can be applied directly to any data in binary format, the special properties of language data are akin to cause problems due to the size of the symbol alphabet. Stibor et al. [12] have 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. under 32 characters). This limitation is problematic since we are interested in analyzing variable-length text segments consisting of thousands of characters. Also, most of the existing NSA matching rules assume either a binary string representation or a symbol alphabet significantly smaller than in any writing system of natural languages.

A review by Ji and Dasgupta [13] presents three domain-specific customizations which are needed for applying the basic NSA into new areas: (i) a suitable data representation scheme (ii) a similarity measure for data items and (iii) an efficient method for generating detector strings. Previously [14], the NSA has been applied to language

data using a bag-of-words matching rule. In the following, these three are considered for written language on a lower, first-order statistical level.

3.1. Negative representations of language

While the sparsity and large variance in texts are usually a problem for language modeling, these same properties can be useful in terms of applying negative selection to language. Namely, if we select an approximate matching rule for strings, it will be easy to find strings which do not match a specific document (self) while still having a high probability of matching other text segments (non-self).

The aim of the subsequent process is to produce a collection of detectors which is a) as compact in size as possible, b) as likely as possible to match the document after any change c) does not match document D for which the detectors are generated. Using the first-order language model—the mere frequency of individual characters—we could thus consider the string ‘e’ as the first candidate as it is the most common single-character string in English.¹ Instead of matching two strings directly, we can remove the significance of the character order and focus merely on the frequency of each character in a segment of text. Effectively this means transforming strings into bag-of-characters (i.e. character multisets). Negative information on the document D can be expressed as a sequence of w adjacent characters in the document which does *not* contain a certain proportion of character c .

When considering the sparsity of observed character sequences we can compare the theoretical upper limits of unique character sequences and sets and compare it to a large corpus. In Figure 2 the theoretical upper limits for unique strings and characters of w characters along with the number of observed unique items in the Reuters corpus [15].

3.2. Character frequency based matching rule

To avoid evaluating the universe of w^m permutations of w -length strings from a symbol alphabet of m characters, we can simplify the matching rule by considering strings as character multisets. We can now apply the observation about the high variance of the character frequency distributions of short text segments and use this property to find character frequency properties that are absent in some specific document. Instead of computing the mutual distance between two strings, we compute the character frequency histograms of all of the w -length substrings of the document and match it with detectors that are defined as specific frequencies of a given character using some fixed window length.

Using the Text-NSA matching scheme the NSA concepts of a detector and the matching rule have the following interpretation:

¹As a side note the 1939 novel “Gadsby” by E. V. Wright, however, is an anecdotal example of a complete book which does not contain a single ‘e’.

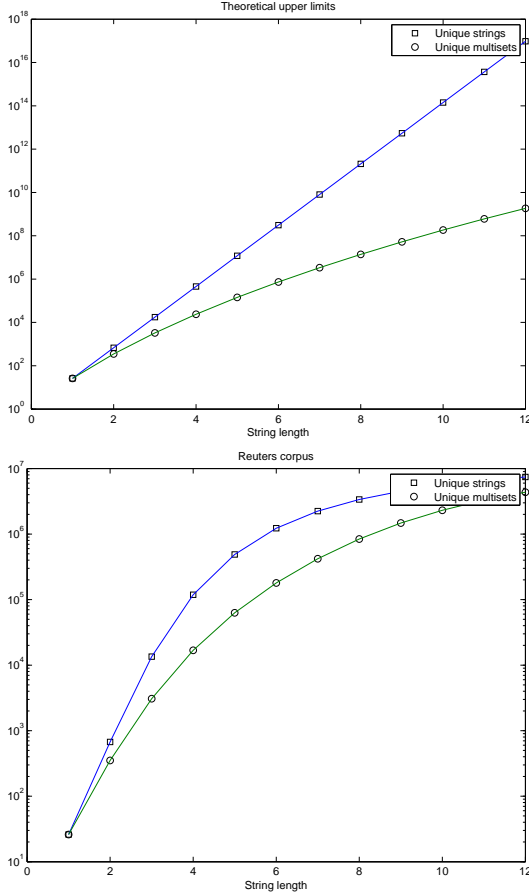


Figure 2. Theoretical upper limits (top) for the number of unique w -length character permutations and character multisets and the observed quantities in the Reuters corpus (bottom).

Detector: A Text-NSA detector is defined as a character c , a window length parameter w and a specific frequency k denoting the amount of occurrences of c .

Matching rule: A match between a detector and a passage of text (consisting of l characters $c_1 \dots c_l$) is positive if there exists a position i in the text where the character c appears exactly k times in the sequence of characters $\{c_i, c_{i+1}, \dots, c_{i+w-1}\}$.

Essentially, the detector collection is thus a negative first-order description of the character statistics of a document.

3.3. Detection probabilities

The Bernoulli trial can be used to analyze the probabilities of detecting a random change in a text segment. For simplicity, we use the first-order language model i.e. the assumption of text being a sequence of randomly and independently generated characters according to a character probability distribution $p(c_1, c_2, \dots, c_m)$.

The probability of observing k occurrences of character c in a sequence of w characters is given by the Bino-

mial distribution

$$\text{Bin}(k|w, p) = \binom{w}{k} p^k (1-p)^{w-k} \quad (2)$$

where p is the probability of the first-order model for character c . We can use this information in constructing a first-order model for character frequencies which are not observed in a text document by considering the probability that a sequence of w characters does not contain k occurrences of c

$$p(\bar{k}|w, p) = 1 - \binom{w}{k} p^k (1-p)^{w-k} \quad (3)$$

This is a useful result for organizing the collection of detectors according to the likeliness of matching any segment of text. In the change detection phase this information can be used to speed up the algorithm such that the most probable detectors can be matched first.

For example, consider a passage of text for which all the 9-character long substrings contain either two or three 'e's. Assuming $p = 0.12$ the probability of a random sequence of 9 characters with such feature is

$$\begin{aligned} p(k = \{2, 3\} | w = 9, p = 0.12) \\ = \sum_{i=\{2,3\}} \binom{9}{i} 0.12^i (1 - 0.12)^{9-i} = 0.279. \end{aligned} \quad (4)$$

We can use this information to detect changes by generating detectors for $w = 9, k = \{0, 1, 4, 5, 6, 7, 8, 9\}$. If the original string would be replaced with 9 random characters, the probability of detection would be

$$\begin{aligned} p(k = \{0, [4, 9]\} | w = 9, p = 0.12) \\ = \sum_{i=\{0,1\}, [4,9]} \binom{9}{i} 0.12^i (1 - 0.12)^{9-i} = 0.721 \end{aligned} \quad (5)$$

When combining this with the detector information for the remaining 25 characters, the probability of detecting a random change increases even further.

In terms of optimizing the detection probability, the choice of the window length parameter w has a strong effect. The total amount of unique character multisets of cardinality w is given by the multiset coefficient

$$\binom{m+w-1}{w} \quad (6)$$

where m is the size of the symbol alphabet. For window lengths $[1, w]$ the size of the initial detector population R_0 then becomes

$$N_{R_0} = \sum_{i=1}^w \binom{m+i-1}{i} \quad (7)$$

From a string of l characters we can extract $l - w + 1$ substrings of length w and compute the character frequency histogram for each of them. The frequency histogram data becomes increasingly sparse when the window length w is increased resulting in a situation where

a small change in the string is likely to change the frequency histogram, but also the number of detectors to be generated becomes large. Thus, there is a tradeoff between highly sensitive detection and a compact detector collection. In the following experiment we study this dependence as a balance between the accuracy of the result and the selected w .

4. EXPERIMENTS ON A WIKIPEDIA ARTICLE COLLECTION

4.1. Experiment setting

A collection of 200 randomly selected Wikipedia [16] articles was used as a test set for an experiment in analyzing changes in text documents. Two versions of the articles were downloaded—one version from June 2007 and a new version from March 2008 after being freely editable for nine months.

Preprocessing was applied to the articles in order to remove redundant information in terms of the article contents. To focus the analysis solely on content a conversion to lowercase letters was performed and all whitespace and punctuation characters were removed. Thus after the preprocessing stage the articles were truncated into strings of characters [a-z] with their length varying from 1001 to 56 550 characters.

As a baseline result for the detection task the Levenshtein distance (edit distance) between the original and the updated version of the article was computed. The Levenshtein distance of two strings is the smallest amount of atomic modifications (adding/removing/replacing a single character) to transform one string into the other one. Of the 200 articles 122 (61 %) contained modifications ranging from 2 to 4840 in Levenshtein distance.² The remaining 78 articles had not been updated. The median of the edit distances was 53 and the mean was 299.3 atomic edits. A summary of the properties of the used corpora including the article sizes is presented in Table 1.

To detect the changes in the article collection, a population of detector strings was generated for each string describing a window length w , a character c for which the detector was used and a target frequency which was not observed in the original document. In this experiment we generated an exhaustive collection of all detectors of length 1–250 for all 26 letters of the Latin alphabet. All of these were matched against the original article version and the matching ones were removed.

In the change detection phase each of the detectors were matched against the updated article versions. The interest was especially in finding out the minimum window length for gaining a complete accuracy.

4.2. Results

Two examples of successful results can be found in Figure 3 where the non-self matching detectors (gray dots) and the detectors matching the changed versions (black

²For the longest three articles the distance could not be computed using the standard algorithm due to memory limitations.

Table 1. Statistics of the Wikipedia article corpus ($N = 200$) used in the experiment (character count).

	mean	median	min	max
Original article	3 952	2 607	1 001	56 550
Updated article	3 965	2 582	1 001	54 352
Difference	12.67	0	-2 836	2 451
L-distance	299.3	53	0	4 840

squares) are shown for a narrow ($w = 4$) and a wide ($w = 163$) window length. As seen in the figure, a change in the article “Panettone” was detected by the frequency of characters ‘c’, ‘o’ and ‘r’ which matched the detector for $k = 3$. In the lower figure, a detector for a low proportion of character ‘s’ is activated for the article “Phylocode”.

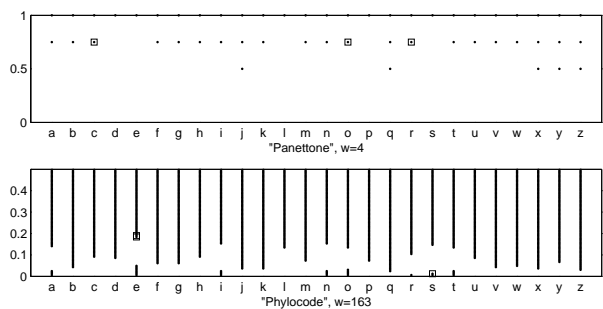


Figure 3. Detection of changes using a small and large window lengths. In the upper figure, a change is detected for three characters in the article “Phylocode” using a window of four characters. In the lower figure, a lower frequency limit for ‘s’ is detected in the article “Panettone”.

In Figures 4 and 5 the detection rate (i.e. accuracy of the classification) is shown as a function of the used window length. The detection rate is calculated as

$$\begin{aligned} \text{detection rate} &= \frac{\text{non-self correctly classified}}{\text{total non-self}} \\ &= \frac{\text{TP}}{\text{TP} + \text{FN}} \end{aligned} \quad (8)$$

where TP (true positives) is the detected number of documents which contain modifications and FN (false negatives) is the number of documents that contain changes that have not been detected. While the absence of false positives is guaranteed due to the matching algorithm, the detection rate (sensitivity) is an appropriate measure for the quality of the result.

As seen in Figure 4 the first-order statistics of a mere 20 subsequent characters was enough to detect half of the changes and a quarter of the changes was detected with only $w = 4$. In Figure 5 the same result is shown with a normalized window length \hat{w} which is scaled to the length of the original document to analyze the relationship of the required w in comparison to the length of the document. In Figure 6 the dependency between the window length and the edit distance is shown on a logarithmic scale. In this figure, each modified article is shown as a dot such

that a dot in the left bottom corner indicates a small modification which was detected with a small value of w and points in the upper right segment represent inefficient detection results where a large window length was needed.

The main result of the Wikipedia experiment is the fact that a detector window length of 210 characters was enough to detect all changes in the article corpus while the median of the required window length was only 20 characters (0.47% of the document length). A summary of the required window lengths is presented in Table 2.

Table 2. Results of detecting the edited articles with varying window lengths w .

w	Detected documents	Detection rate	\hat{w}	Detected documents	Detection rate
1	2	0.01	0.02 %	6	0.05
2	18	0.15	0.03 %	10	0.08
3	22	0.18	0.04 %	11	0.09
4	30	0.25	0.05 %	11	0.09
5	32	0.26	0.06 %	13	0.11
6	41	0.34	0.07 %	14	0.11
7	49	0.40	0.08 %	17	0.14
8	51	0.42	0.09 %	20	0.16
9	51	0.42	0.1 %	24	0.20
10	52	0.42	0.2 %	40	0.32
20	62	0.51	0.4 %	55	0.45
40	92	0.75	0.8 %	74	0.61
60	105	0.86	1.2 %	85	0.70
90	110	0.90	1.6 %	91	0.75
120	112	0.91	2 %	98	0.80
150	114	0.93	6 %	119	0.98
180	120	0.98	10 %	121	0.99
210	122	1.00	16 %	122	1.00

In Figure 4 the detection rate is shown as a function of the used maximum detector length. In Figure 5 the window lengths have been normalized for the length of the original document.

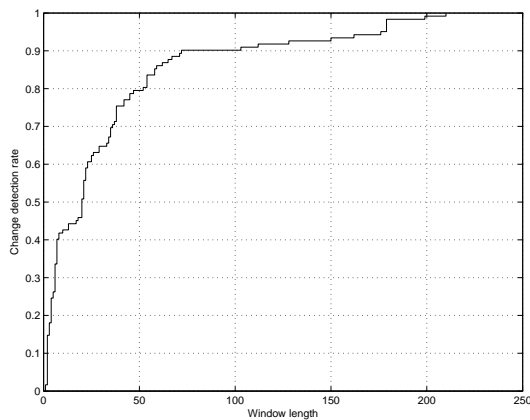


Figure 4. Detection rate as a function of the window length. An accuracy of 50 % is obtained with $w = 20$.

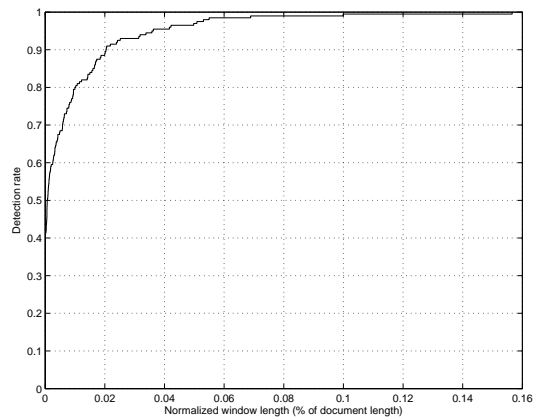


Figure 5. Detection rate with normalized window lengths. A detector window length of 2 % of the original document length is enough to detect 90 % of the changes.

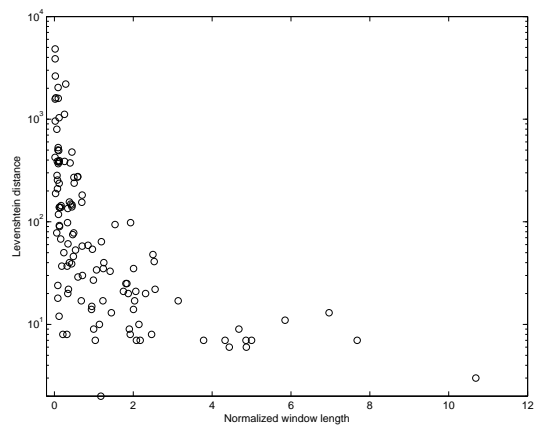


Figure 6. The relation between the required normalized window length (horizontal axis) and the Levenshtein distance of the edits (vertical axis). Each point represents a single article.

5. DISCUSSION AND FUTURE WORK

We have presented a probabilistic method for analyzing changes between two text documents as an alternative to traditional tools based on cryptographic hash functions. Our approach is motivated by biological immune systems and the consequent beneficial properties of distributability and fault tolerance motivate its use in new application areas. Text mining and the analysis of large text corpora on web sites are an example of such.

The characteristic features of the presented change detection algorithm are the ability to detect changes without explicit knowledge about the original document which is a valuable property in terms of protecting the privacy of the original data [17]. Also, the nature of the detection process allows the process to be run in parallel using distributed computation nodes. In addition to distributing the detector collection into several nodes, the analyzed document can itself be divided into parts since any part of the document can be analyzed independently.

As a contrast to traditional integrity analysis methods which are based on computing cryptographic hash functions, we presented the ability to detect the location of the change as a novel feature. The location of the detected change can be tracked to all subparts of the document where any non-self detector has a match. Also, the number of matching detectors gives a rough estimate on the magnitude of the changes.

The theoretical discussion on the detection probability and the experiments of Section 4 were presented in the limited scope of a 26-character Latin alphabet (from a to z). The theoretical analysis of Section 3.3 gives us reason to expect improved detection results when using larger symbol vocabularies (including case-sensitivity and punctuation in the analysis). For writing systems with significantly larger grapheme vocabularies (extended Latin alphabet or eastern writing systems) we expect even better results as a consequence of the sparsity of the data.

6. ACKNOWLEDGMENT

This research work was funded by the Academy of Finland under Grant 214144.

7. REFERENCES

- [1] Andrew Tridgell, *Efficient Algorithms for Sorting and Synchronization*, Ph.D. thesis, Australian National University, 1999.
- [2] Giovanni Di Crescenzo and Faramak Vakil, "Cryptographic hashing for virus localization," in *WORM'06: Proceedings of the 4th ACM Workshop on Recurring Malcode*, New York, NY, USA, 2006, pp. 41–48, ACM.
- [3] Konstantinos Stamatakis, Konstantinos Chandrinou, Vangelis Karkaletsis, Miquel Angel Mayer, Dagmar Villarroel Gonzales, Martin Labský, Enrique Amigó, and Matti Pöllä, "AQUA, a system assisting labelling experts assess health web resources," in *Proceedings of 12th International Symposium for Health Information Management Research, iSHIMR 2007*, Sheffield, UK, July 2007, Accepted for publication.
- [4] Konstantinos Stamatakis, Vangelis Metsis, Vangelis Karkaletsis, Marek Růžička, Vojtech Svátek, Enrique Amigó Cabrera, and Matti Pöllä, "Content collection for the labeling of health-related web content," in *Proceedings of 11th Conference on Artificial Intelligence in Medicine (AIME 07)*, Amsterdam, The Netherlands, July 2007, Accepted for publication.
- [5] Charles Reis, Steven D. Gribble, and Tadayoshi Kohno, "Detecting in-flight page changes with web tripwires," in *Proceedings of NSDI'08: 5th USENIX Symposium on Networked Systems Design and Implementation*, 2008, pp. 31–44.
- [6] Stephanie Forrest, Alan S. Perelson, Lawrence Allen, and Rajesh Cherukuri, "Self-nonsel self discrimination in a computer," in *Proceedings of the 1994 IEEE Symposium on Research in Security and Privacy*, Oakland, CA, 1994, pp. 202–212, IEEE Computer Society Press.
- [7] Patrick D'haeseleer, Stephanie Forrest, and Paul Helman, "An immunological approach to change detection: algorithms, analysis, and implications," in *Proceedings of the Symposium on Research in Security and Privacy*, May 1996, pp. 110–119, IEEE Computer Society Press.
- [8] Patrick D'haeseleer, "An immunological approach to change detection: theoretical results," in *Proceedings of the 9th Computer Security Foundations Workshop*, 1996, pp. 18–26, IEEE Computer Society Press.
- [9] Leandro N. de Castro and Jonathan Timmis, Eds., *Artificial Immune Systems: A New Computational Intelligence Approach*, Springer-Verlag, 2002.
- [10] Stephanie Forrest, Steven A. Hofmeyr, Anil Somayaji, and Thomas A. Longstaff, "A sense of self for Unix processes," in *Proceedings of the 1996 IEEE Symposium on Research in Security and Privacy*, 1996, pp. 120–128, IEEE Computer Society Press.
- [11] D. Dasgupta and S. Forrest, "Tool breakage detection in milling operations using a negative-selection algorithm," 1995.
- [12] Thomas Stibor, Philipp Mohr, Jonathan Timmis, and Claudia Eckert, "Is negative selection appropriate for anomaly detection?," in *GECCO '05: Proceedings of the 2005 conference on Genetic and evolutionary computation*, New York, NY, USA, 2005, pp. 321–328, ACM.

- [13] Zhou Ji and Dipankar Dasgupta, “Revisiting negative selection algorithms,” *Evolutionary Computation*, vol. 15, no. 2, pp. 223–251, 2007, (Summer 2007).
- [14] Matti Pöllä and Timo Honkela, “Probabilistic text change detection using an immune model,” in *Proceedings of the International Joint Conference on Neural Networks, IJCNN 2007*, Orlando, Florida, August 2007, pp. 1109–1114.
- [15] “Reuters corpus, volume 1, english language, 1996-08-20 to 1997-08-19, release date 2000-11-13, format version 1,” <http://about.reuters.com/researchandstandards/corpus/>, 2000.
- [16] “Wikipedia, the free encyclopedia,” <http://en.wikipedia.org/wiki/>.
- [17] Fernando Esponda, Elena S. Ackley, Paul Helman, Haixia Jia, and Stephanie Forrest, “Protecting data privacy through hard-to-reverse negative databases,” *International Journal on Information Security*, vol. 6, no. 6, pp. 403–415, 2007.