Towards Intrusion Detection by Information Retrieval and Genetic Programming

Pavel Krömer, Jan Platoš,Václav Snášel
VŠB - Technical University of Ostrava
17. listopadu 15, 708 33
Ostrava-Poruba, Czech Republic
Email:{pavel.kromer, vaclav.snasel, jan.platos}@vsb.cz

Ajith Abraham
Machine Intelligence Research Labs (MIR Labs)
Scientific Network for Innovation and Research Excellence,
WA, USA
Email: ajith.abraham@ieee.org

***Abstract*—Fuzzy classifiers and fuzzy rules are powerful tools in data mining and knowledge discovery. In this work, intrusion detection is approached as a data mining task and genetic programming is deployed to evolve fuzzy classifiers for detection of intrusion and security problems. We train the fuzzy classifier on a data set modeled as a fuzzy information retrieval collection and investigate its ability to detect illegitimate actions. Proposed approach is experimentally evaluated on the popular KDD Cup intrusion detection data set.**

I. Introduction

Genetic programming is a powerful machine learning technique from the wide family of evolutionary algorithms. In contrast to traditional evolutionary algorithms, it can be used to evolve tree structures and symbolic expressions. It has been used to evolve Lisp S-expressions, mathematical functions, symbolic expressions, decision trees and recently to infer search queries describing relevance ranked documents in an fuzzy information retrieval system.

The last application seems to be interesting for general data mining and applications since it can be directly applied to data mining problems. Extended Boolean queries (e.g. fuzzy queries) from information retrieval can be interpreted as flexible fuzzy classifiers that describe a fuzzy subset of data by means of its features. Fuzzy classifiers evolved over a training data set can be subsequently used to classify incoming data samples and hence they can serve e.g. as a part of an intrusion detection system to detect security threats and intrusions.

Intrusion detection systems (IDS) were proposed to prevent security violations in computer systems and networks [2], [11]. They are intended to reinforce basic security measures such as authentication and access control and detect actions that aim to break security policies in protected computer system. Intrusion detection can be defined as an identification of users, hosts, or programs that are using a system resources without authorization. Moreover, authorized system users that are abusing their privileges are subject to intrusion detection as well. In the rest of this paper, we introduce and investigate an intrusion detection approach based on information retrieval and utilizing genetic programming. In section II, information retrieval, genetic programming, and the concept of evolutionary query optimization is briefly introduced. In section III, we describe the evolution of an classifier for intrusion detection and its experimental evaluation.

II. Preliminaries

The evolution of fuzzy classifiers for intrusion detection is implemented in the framework for search query optimization. Data samples are interpreted as documents, and features are mapped to index terms. We briefly introduce basic concepts of fuzzy information retrieval. Next, we provide short description of the genetic programming.

A. Fuzzy information retrieval

The area of information retrieval (IR) is a branch of computer science dealing with storage, maintenance, and searching in large amounts of data. The data could be in different formats, e.g. textual, visual, audio, or multimedia documents [3].

An information retrieval system (IRS) is a software tool serving for data representation, storage and subsequent information searching. The amount of documents contained in data collections managed by an IRS is usually very large and the task of easy, efficient, and accurate information searching is especially challenging.

An IR model is a formal background defining the internal document representation, query language, and a document – query matching mechanism. Consequently, the model determines the document indexing procedure, result ordering, and other aspects of a particular information retrieval system. In this study, we have implemented extended Boolean IR model.

The extended Boolean model of IR is based on fuzzy set theory and fuzzy logic [3], [7]. Documents are interpreted as fuzzy sets of indexed terms, assigning to every term contained in the document a particular weight from the range $[0, 1]$ expressing the degree of significance of the term for document representation. A formal collection description in the extended Boolean IR model is shown in eq. (1) and eq. (2), where d_i represents $i-$th document and t_{ij} $j-$th term in $i-$th document. An index matrix of the entire document collection is denoted $\boldsymbol{D}$.

$$\boldsymbol{d}_i = (t_{i1}, t_{i2}, \ldots, t_{im}),\ \forall\ t_{ij} \in [0,1] \tag{1}$$

$$\boldsymbol{D} = \begin{pmatrix} t_{11} & t_{12} & \cdots & t_{1m} \\ t_{21} & t_{22} & \cdots & t_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ t_{n1} & t_{n2} & \cdots & t_{nm} \end{pmatrix} \tag{2}$$

Appropriate indexing procedure is essential for the exploitation of the benefits of the extended Boolean IR model. The internal documentary collection model should be an accurate snapshot of the collection of text documents in natural language and at the same time a basis for an efficient and practical search. In [7] Kraft proposed the usage of Salton's $tf \cdot idf_t$ indexing formula introduced for the vector space model of IRS as a document indexing mechanism in the extended Boolean IR model

The query language in the extended Boolean model of IR is upgraded by the possibility of weighting query terms in order to attribute different levels of importance to those in a search request and by weighting (parameterizing) aggregation operators to soften or blur their impact on query evaluation [3], [7]. Consider Q to be the set of user queries over a collection; then the weight of term t in query q is denoted as $a(q,t)$ satisfying $a : Q \times T \to [0,1]$. To evaluate the atomic query of one term, therefore stating only one search criterion, the function $g : [0,1] \times [0,1] \to [0,1]$ will be used. The value of $g(F(d,t),a)$ is called he retrieval status value (RSV). For RSV enumeration the interpretation of the query term weight a is crucial. The most commonly used interpretations see the query term weight as the importance weight, threshold or ideal document description [3], [7].

In this study, we adopt the threshold interpretation defined in eq. (3) and illustrated in fig. 1. The functions $P(a)$ and $Q(a)$ are coefficients used for tuning the threshold curve. An example of $P(a)$ and $Q(a)$ could be as follows: $P(a) = \frac{1+a}{2}$ and $Q(a) = \frac{1-a^2}{4}$. Acording to the threshold interpretation, an atomic query containing term t of the weight a is a request to retrieve documents having $F(d,t)$ equal or greater to a. For documents satisfying this condition will be rated with high RSV and contrariwise documents having $F(d,t)$ smaller than a will be rated with small RSV.

$$g(F(d,t),a) = \begin{cases} P(a)\frac{F(d,t)}{a} & \text{for } F(d,t) < a \\ P(a) + Q(a)\frac{F(d,t)-a}{1-a} & \text{for } F(d,t) \geq a \end{cases} \tag{3}$$

Precision P and recall R are among the most used IR effectiveness measures. They are defined in eq. (4), where REL stands for the set of all relevant documents and RET for the set of all retrieved documents. Precision can be then understood as the probability of retrieved document to be relevant and recall can be seen as the probability of retrieving relevant document.

$$P = \frac{|REL \bigcap RET|}{|RET|} \qquad R = \frac{|REL \bigcap RET|}{|REL|} \tag{4}$$

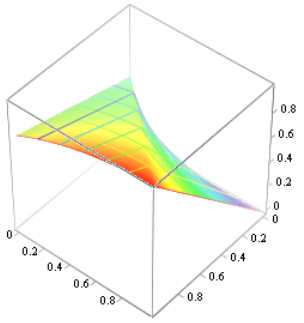


Fig. 1: $g(F(d,t),a)$ according to (3).

Precision and recall in the extended Boolean IR model can be defined using sigma count $\|A\|$ [8]:

$$\rho(X|Y) = \begin{cases} \frac{\|X \cap Y\|}{\|Y\|} & \|Y\| \neq 0 \\ 1 & \|Y\| = 0 \end{cases} \tag{5}$$

$$P = \rho(REL|RET) \qquad R = \rho(RET|REL) \tag{6}$$

For easier IR effectiveness evaluation were developed measures combining precision and recall into one scalar value. F-score F [9] is among the most used scalar combinations of P and R.

$$F = \frac{(1+\beta^2)PR}{(\beta^2 P + R)} \tag{7}$$

In the F-score as defined in eq. (7) (according to van Rijsbergen), β allows to prioritize P or R.

B. Genetic algorithms and genetic programming

Genetic algorithms are a popular member of the wide chapter of evolutionary algorithms. They are based on the programmatical implementation of genetic evolution and they emphasize selection and crossover as the most important operations in the evolutionary optimization process [5], [10].

Genetic algorithms evolve a population of chromosomes representing potential problem solutions encoded into suitable data structures. The evolution is performed by genetic operators modifying the chromosomes, i.e. the encoded forms of problem solutions. Proper encoding is vital for the effectiveness of the evolutionary searches. It defines the genotype, the space of all encoded problem solutions, which is different from the phenotype, the space of all problem solutions. Genetic algorithms explore the genotype of the problem being investigated and the size and shape of the problem genotype define its fitness landscape.

Genetic programming is an extension to genetic algorithms, allowing work with hierarchical, often tree-like, chromosomes with an unlimited length [5], [6].

Genetic programming was introduced as a tool to evolve whole computer programs and represented a step towards adaptable computers that could solve problems without being programmed explicitly [1]. This is an important ability because solutions to most problems can be formulated by the means

1 Define objective (fitness) function and problem encoding
2 Encode initial population P of possible solutions as fixed length strings
3 Evaluate chromosomes in initial population using objective function
4 **while** *Termination criteria not satisfied* **do**
5 Apply selection operator to select parent chromosomes for reproduction: $sel(P_i) \rightarrow parent_1$, $sel(P_i) \rightarrow parent_2$
6 Apply crossover operator on parents with respect to crossover probability P_C to produce new chromosomes: $cross(P_C, parent_1, parent_2) \rightarrow \{offspring_1, offspring_2\}$
7 Apply mutation operator on offspring chromosomes with respect to mutation probability P_M: $mut(P_M, offspring_1) \rightarrow offspring_1$, $mut(P_M, offspring_2) \rightarrow offspring_2$
8 Evaluate offspring chromosomes: $fit(offspring_1) \rightarrow offspring_1^{fit}$, $fit(offspring_2) \rightarrow offspring_2^{fit}$
9 Create new population from current population and offspring chromosomes: $migrate(offspring_1, offsprig_2, P_i) \rightarrow P_{i+1}$
10 **end**

Algorithm 1: A summary of genetic algorithm

of computer programs. Moreover, genetic programming can be used to develop solutions in the field of machine learning, symbolic processing, or any other domain that can formulate its solutions by means of parseable symbolic expression. Genetic programming allows the efficient evolution of such symbolic expressions with well-defined syntax and grammar.

The birth of modern genetic programming is attributed to John Koza [5]. The introduction of genetic programming was followed by extensive theoretical work, studying why genetic programming actually works, and by a number of success stories that attracted attention to this sort of evolutionary algorithms [1].

In GP the chromosomes take the form of hierarchical variably-sized expressions, point-labeled structure trees. The trees are constructed from nodes of two types, terminals and functions. More formally, a GP chromosome is a symbolic expression created from terminals t from the set of all terminals T and functions f from the set of all functions F satisfying the recursive definition [1]:

1) $\forall t \in T : t$ is the correct expression
2) $\forall f \in F : f(e_1, e_2, \ldots, e_n)$ is the correct expression if $f \in F$ and $e + 1, \ldots, e_n$ are correct expressions. The function $arity(f)$ represents the arity of f.
3) there are no other correct expressions

GP chromosomes are evaluated by the recursive execution of instructions corresponding to tree nodes [1]. Terminal nodes are evaluated directly (e.g. by reading an input variable) and functions are evaluated after left-to-right depth-first evaluation of their parameters.

Genetic operators are applied to the nodes in tree-shaped chromosomes. A crossover operator is implemented as the mutual exchange of randomly selected subtrees of the parent chromosomes. For an example see fig. 2. Mutation has to modify the chromosomes by pseudorandom arbitrary changes in order to prevent premature convergence and broaden the coverage of the fitness landscape. Mutation could be implemented as:

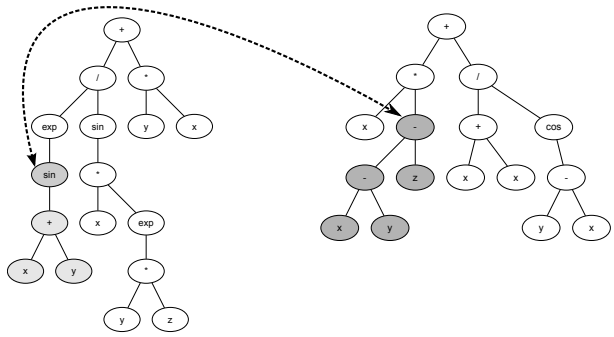


Fig. 2: Crossover in genetic programming.

- removal of a subtree at a randomly chosen node
- replacement of a randomly chosen node by a newly generated subtree
- replacement of node instruction by a compatible node instruction (i.e. a terminal can be replaced by another terminal, a function can be replaced by another function of the same arity)
- a combination of the above

Genetic programming facilitates the efficient evolution of symbolic expressions, even whole computer programs. In this work, we use genetic programming for fuzzy classifier optimization.

C. Evolutionary query optimization

Genetic programming has been recently used for the optimization of extended Boolean queries [4], [12]. It was shown that genetic programming was able to optimize search queries so that they described a set of relevant documents. In the fuzzy information retrieval model, the relevant documents formed a fuzzy subset of the set of all documents and the extended Boolean queries were evolved to describe them.

An information retrieval system based on the extended Boolean IR model was implemented to validate evolutionary query optimization. The $tf \cdot idf_t$ term statistics were used for document indexing and query weights (RSV) were evaluated using eq. (3). The query language in the IRS supported the standard Boolean operators AND, OR, and NOT.

The information retrieval system served as a test bed for evolutionary query optimization and allowed genetic programming over extended Boolean queries. The GP evolved tree representations of search queries with Boolean operators as function nodes and terms as leaves. Both operator nodes and term nodes were weighted. In order to generate a random initial population for the GP, the system was able to generate random queries. The particular settings of the random query generator showing the probabilities of generating a particular query node are summarized in table Ia. An example of three random queries generated by the system is shown in fig. 3.

We have seen in section II-B that the implementation of a crossover operator for GP is straightforward. In the experimental information retrieval system, it was implemented

as a mutual exchange of two randomly selected branches of parent tree chromosomes. The mutation operator in query GP aims to perturb the content and structure of the chromosomes randomly. In our implementation, it selects a node from the processed chromosome at random and performs one of the mutation operations summarized in table Ib.

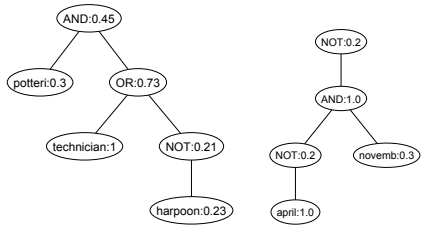
TABLE I: Random query generation an mutation probabilities.

(a) Probabilities of generating random query nodes.

Event	Probability
Generate term	0.5
Generate operator AND	0.24
Generate operator OR	0.24
Generate operator NOT	0.02

(b) Probabilities of mutation operations.

Event	Probability
Mutate node weight	0.5
Insert or delete NOT node	0.1
Replace with another node or delete NOT node	0.32
Replace with random branch	0.08



(a) Query potteri:0.3 AND:0.45 (technician OR:0.73 NOT:0.21 harpoon:0.23).

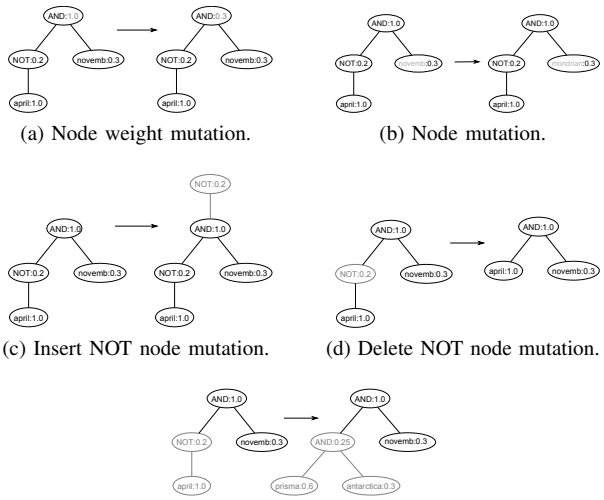
(b) Query NOT:0.2 ((NOT:0.2 april) AND novemb:0.3).

(c) Query (prisma:0.6 AND:0.25 antarctica:0.3) OR:0.33 (orlov:0.3 AND:0.22 sharp:0.1).

Fig. 3: Example of random queries.

The query mutation types that were implemented included:

- change of selected node weight. This mutation type is shown in fig. 4a
- replacement of selected node type by a compatible node type (i.e. operator OR replace by operator AND, term replaced by another term). This mutation type is shown in fig. 4b.
- insertion of NOT operator before selected node (fig. 4c).
- removal of NOT operator if selected (fig. 4d).
- replacement of selected node by a randomly generated branch (fig. 4d).



(a) Node weight mutation.

(b) Node mutation.

(c) Insert NOT node mutation.

(d) Delete NOT node mutation.

(e) Replace node by branch mutation.

Fig. 4: Query mutation types.

The well-known IR measure F-Score was used as a fitness function.

The extended queries evolved by the algorithm can be seen as fuzzy classifiers describing the fuzzy set of relevant documents. The fuzzy classifier evolved over a training collection of documents, or more general data records, can be used to classify new documents (data samples).

III. Genetic evolution of fuzzy classifier for intrusion detection

The algorithm for evolutionary query optimization was applied to the evolution of a symbolic fuzzy classifier. In this study, we have implemented an evolution of fuzzy classifier for intrusion detection.

A. KDD Cup 1999 data set

To investigate the ability of discussed algorithm to find useful classifiers, a test sytsem implementing evolution of fuzzy expressions was implemented. The 10% sample of the KDD Cup 1999 intrusion detection dataset[1] was used to evolve classifiers and test their ability to detect illegal actions. It contains 10% of the large intrusion detection data set created in 1998 by the DARPA intrusion detection evaluation program at MIT. The full data set contains 744 MB data with 4,940,000 records with 41 nominal and numerical features. For our experiments, all features were converted to numeric and subsequetly normalized.

The data describes normal traffic and 4 attack classes called DoS (Denial of Services), U2R (User to Root), R2L (Remote to User), and Probe (Probing). The records for each class are divided into training (40%) and testing (60%) data set. For each class, the training data set was used to evolve the fuzzy classifier and testing data set was used to evaluate the detection capabilities of the classifier. The attack classes contained

[1] http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

following number of records: DoS contained 195,494 training and 293,242 testing records, U2R consisted of 38,931 training and 58,399 testing records, R2L included 39,361 training and 59,043 testing records, and finally the Probe class consisted of 40,553 training and 60,832 testing records.

B. Intrusion detection classifier evolution

The evolution of an intrusion detection classifier differs from the query optimization task only semantically. We interpret data samples as documents and features as terms. The normalized feature value corresponds to the index weight of a term in a document (feature weight in a data sample) while the class of the record corresponds to document relevance. In the testing data, there are only 2 crisp product classes: normal traffic (class 0) and attack (class 1). The goal of the algorithm was to find an expression (fuzzy classifier) that will describe the set of records describing an attack. The mapping of normalized data onto an IRS index matrix is illustrated in table IIb.

TABLE II: KDD Cup intrusion detection data set.

(a) Normalized features

Id	Feat. 1	Feat. 2	...	Feat. 839	Prod. class
1	0.846	0.951	...	0.148	1
2	0.856	0.9452	...	0.160	1
3	0.882	0.968	...	0.160	0
⋮	⋮	⋮		⋮	⋮

(b) Intrusion detection data set as an IRS index matrix $\boldsymbol{D}$.

$$\boldsymbol{D} = \begin{pmatrix} 0.846 & 0.951 & \cdots & 0.148 \\ 0.856 & 0.9452 & \cdots & 0.160 \\ \vdots & \vdots & \ddots & \vdots \\ 0.618 & 0.861 & \cdots & 0.025 \end{pmatrix}$$

The settings for the GP are summarized in table III.

TABLE III: GA parameters used for fuzzy classifier evolution.

Parameter	Value
Population size	100
Generations limit	5000
Fitness	F-Score according to eq. (7)
Mutation probability	0.8
Crossover probability	0.2

The F-Score parameter β was set in different experiments to 1, 0.5 and 5 to see detection capabilities of evolved classifiers with different priorities of precision and recall in fitness function. We have observed overall accuracy of the classification (OA) as the percent of correctly classified records in the test collection, detection rate (DR), e.g. the percent of correctly classified attacks and false positives (FP), e.g. the percent of regular records missclassified as attacks. Obviously, good classifier would feature high OA, high DR and low FP.

The results of experiments are summarized in table IV.

TABLE IV: Calssification results for different attack classes.

(a) Results for attack class DoS

	β		
	0.5	1	5
OA	93.95	99.31	95.22
DR	99.42	99.27	94.04
FP	28.07	0.53	0.05

(b) Results for attack class U2R

	β		
	0.5	1	5
OA	99.95	99.96	99.95
DR	50	34.34	50
FP	0.02	0	0.02

(c) Results for attack class R2L

	β		
	0.5	1	5
OA	93.95	98.87	99.09
DR	99.42	38.17	31.07
FP	28.07	0.43	0.12

(d) Results for attack class Probe

	β		
	0.5	1	5
OA	94.02	98.46	98.34
DR	90.34	63.2	59.11
FP	5.83	0.05	0.01

We can see that evolved classifier reached reached in all cases and for all attack classes good accuracy above 93 percent. However, DR and FP are for some attack classes not so good. The best combination of high DR and low FP was reached for DoS and $\beta = 1$. The classifier managed to detect 99.27 percent of attacks and misclassified only acceptable 0.53 percent of harmless connections. For the U2R attack class, the best classifiers managed to detect 50 percent of the attacks. In R2L experiment, the classifier evolved with $\beta = 0.5$ reached DR 99.42 percent, but it also misclassified close to 30 percent of harmless connections. The classifiers with low FP managed to detect only 38 and 31 percent of attacks. Finally, the classifiers evolved for Probe attacks managed to detect fair 90 percent of attacks at the cost of 5.83 percent of false positives for $\beta = 0.5$ and around 60 percent of attacks with FP percent below 0.05 for $\beta = 1$ and $\beta = 5$.

The different results for different attack classes suggest that the nature of the fetaures describing the attacks varies and different settings for GP (e.g. the value of β) needs to be used. Moreover, we have seen that high overall acuuracy of classification does not imply good detection rate and low misclassification of legitimate traffic.

IV. Conclusions

We have implemented a genetic programming to evolve fuzzy classifiers for intrusion detection. The intrusion detection problem was reformulated as an information retrieval task and a search query optimization algorithm was used to infer symbolic fuzzy classifier describing the fuzzy set of attacks in the collection describing network traffic.

The evaluation of the algorithm over the KDD Cup 1999 collection has shown that the algorithm is able, for some attack classes, to find good classifiers that can serve as tools for intrusion detection.

The evolution of fuzzy classifier for intrusion detection is an ongoing project with a number of tasks deserve attention in this case. The choice of the best fitness function (are IR measures

really the best fitness function for classifier evolution?) or the interpretation of fuzzy weights in the classifier (is the IR retrieval status value the optimal choice?) are among the most appealing open questions.

ACKNOWLEDGEMENT

This work was supported by the Ministry of Industry and Trade of the Czech Republic, under the grant no. FR-TI1/420.

REFERENCES

[1] Michael Affenzeller, Stephan Winkler, Stefan Wagner, and Andreas Beham. *Genetic Algorithms and Genetic Programming: Modern Concepts and Practical Applications*. Chapman & Hall/CRC, 2009.

[2] Srilatha Chebrolu, Ajith Abraham, and Johnson P. Thomas. Hybrid feature selection for modeling intrusion detection systems. In *Neural Information Processing*, volume 3316 of *Lecture Notes in Computer Science*, pages 1020–1025. Springer Berlin / Heidelberg, 2004.

[3] Fabio Crestani and Gabriella Pasi. Soft information retrieval: Applications of fuzzy set theory and neural networks. In N. Kasabov and R. Kozma, editors, *Neuro-Fuzzy Techniques for Intelligent Information Systems*, pages 287–315. Springer Verlag, Heidelberg, DE, 1999.

[4] Dusan Húsek, Suhail S. J. Owais, Václav Snášel, and Pavel Krömer. Boolean queries optimization by genetic programming. *Neural Network World*, pages 359–409, 2005.

[5] J. Koza. *Genetic programming: A paradigm for genetically breeding populations of computer programs to solve problems*. Technical Report STAN-CS-90-1314, Dept. of Computer Science, Stanford University, 1990.

[6] John R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge, MA, USA, 1992.

[7] D. H. Kraft, F. E. Petry, B. P. Buckles, and T. Sadasivan. Genetic Algorithms for Query Optimization in Information Retrieval: Relevance Feedback. In E. Sanchez, T. Shibata, and L.A. Zadeh, editors, *Genetic Algorithms and Fuzzy Logic Systems*, Singapore, 1997. World Scientific.

[8] Henrik L. Larsen. Retrieval evaluation. In *Modern Information Retrieval course*. Aalborg University Esbjerg, 2004.

[9] Robert M. Losee. When information retrieval measures agree about the relative quality of document rankings. *Journal of the American Society of Information Science*, 51(9):pp. 834–840, 2000.

[10] Melanie Mitchell. *An Introduction to Genetic Algorithms*. MIT Press, Cambridge, MA, 1996.

[11] Jan Platoš, Václav Snášel, Pavel Krömer, and Ajith Abraham. Detecting insider attacks using non-negative matrix factorization. In *IAS*, pages 693–696. IEEE Computer Society, 2009.

[12] Vaclav Snasel, Ajith Abraham, Suhail Owais, Jan Platos, and Pavel Kromer. User profiles modeling in information retrieval systems. In *Emergent Web Intelligence: Advanced Information Retrieval*, Advanced Information and Knowledge Processing, pages 169–198. Springer London, 2010.