The Journal of Systems and Software 86 (2013) 2–11

Contents lists available at SciVerse ScienceDirect

The Journal of Systems and Software

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

Failure prediction based on log files using Random Indexing and Support Vector Machines

Ilenia Fronza [a,*], Alberto Sillitti [a], Giancarlo Succi [a], Mikko Terho [b], Jelena Vlasenko [a]

[a] Center for Applied Software Engineering, Faculty of Computer Science, Free University of Bolzano-Bozen, Italy
[b] Nokia, Visiokatu, 3, FI-33720 Tampere, Finland

ARTICLE INFO

Article history:
Received 18 May 2011
Received in revised form 5 June 2012
Accepted 14 June 2012
Available online 28 June 2012

Keywords:
Failure prediction
Random Indexing
Support Vector Machine (SVM)
Event sequence data
Log files

ABSTRACT

Research problem: The impact of failures on software systems can be substantial since the recovery process can require unexpected amounts of time and resources. Accurate failure predictions can help in mitigating the impact of failures. Resources, applications, and services can be scheduled to limit the impact of failures. However, providing accurate predictions sufficiently ahead is challenging. Log files contain messages that represent a change of system state. A sequence or a pattern of messages may be used to predict failures.
Contribution: We describe an approach to predict failures based on log files using Random Indexing (RI) and Support Vector Machines (SVMs).
Method: RI is applied to represent sequences: each operation is characterized in terms of its context. SVMs associate sequences to a class of failures or non-failures. Weighted SVMs are applied to deal with imbalanced datasets and to improve the true positive rate. We apply our approach to log files collected during approximately three months of work in a large European manufacturing company.
Results: According to our results, weighted SVMs sacrifice some specificity to improve sensitivity. Specificity remains higher than 0.80 in four out of six analyzed applications.
Conclusions: Overall, our approach is very reliable in predicting both failures and non-failures.

1. Introduction

The impact of failures on software systems can be substantial since a failure may have irreversible consequences and a recovery process can require unexpected amounts of time and resources (Adiga et al., 2002; Fulp et al., 2008; Lan et al., 2010). Thus, being able to forecast failures is extremely important, even when failures are inevitable – recovery actions can be started on time or, at least, planned. For these reasons, an increasing attention has been paid to failure prediction and a variety of predictive methods have been presented in the last few years (Zheng et al., 2009). In particular, there has appeared a growing interest in predicting failures from sequential data (Fulp et al., 2008; Li et al., 2007; Mannila et al., 1997; Pandalai and Holloway, 2000; Sampath et al., 1994; Srinivasan and Jafari, 1993). Several data mining and machine learning approaches have been proposed where the dataset to be analyzed consists of a sequence of events (Fulp et al., 2008; Salfner, 2008), and each event has an associated time of occurrence.

Log files represent one example of sequential data available; each entry in these files consists of a message. The information recorded varies from general messages (e.g., user logins and status of the system) to critical warnings about program failures (e.g., network problems, i/o errors, etc.). Forensic analysis of the log files can identify causes of failures. Moreover, the information contained in log files can be used for predicting events (Fulp et al., 2008). A classifier can be applied to associate the sequence of messages that precedes an event with a certain group, for example "fail", thereby producing a prediction. Given labeled training data, a Support Vector Machine (SVM) can determine the maximum hyperplane that divides data into two classes. Aggregate features are often used for SVM-based classification (e.g., the average number of messages during a period of time). However, it is important to exploit the sequential nature of system messages (Fulp et al., 2008).

In this paper we introduce a new approach for predicting failures based on SVMs and Random Indexing (RI). Unlike other applications of SVM classifiers (Liang et al., 2007; Xue et al., 2007; Yamanishi and Maruyama, 2005), we use RI to represent sequences of operations (i.e., events) extracted from log files of applications. Then, a SVM classifies the representations provided by RI as either fail or non-fail.

The paper is structured as follows: in Section 2 we discuss existing works in this area; in Section 3, we present some background; in Section 4, we introduce our approach; in Section 5, we describe

* Corresponding author.
E-mail addresses: Ilenia.Fronza@unibz.it (I. Fronza), Alberto.Sillitti@unibz.it (A. Sillitti), Giancarlo.Succi@unibz.it (G. Succi), Mikko.J.Terho@nokia.com (M. Terho), Jelena.Vlasenko@stud-inf.unibz.it (J. Vlasenko).

0164-1212/$ – see front matter © 2012 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.jss.2012.06.025

our experiments and results; in Section 6 we discuss our results and the generalizability of our approach; in Section 7 we introduce our future work.

2. Related work

Methods for predicting failures based on events (e.g., the log messages) have been proposed in various engineering disciplines. These methods can be classified into (1) design-based methods, and (2) data-driven rule-based methods. In design-based methods the expected event sequence is obtained from system design and it is compared with an observed event sequence (Pandalai and Holloway, 2000; Sampath et al., 1994; Srinivasan and Jafari, 1993). The major disadvantage of these methods is that in many cases events occur randomly, thus, there is no information available about the system logic. Data-driven rule based methods do not require any system logic design information. These methods include two phases: (1) identification of temporal patterns, i.e., sequences of events that frequently occur (Mannila et al., 1997), and (2) development of prediction rules based on these patterns (Li et al., 2007).

Several approaches have been proposed for predicting failures using system log files. Such prediction methods include standard machine learning techniques such as Bayes networks, Hidden Markov Models, and Partially Observable Markov Decision Process (Fu and Xu, 2007a,b; Fulp et al., 2008; Gross et al., 2002; Liang et al., 2007; Salfner, 2008; Stearley and Oliner, 2008; Xue et al., 2007; Yamanishi and Maruyama, 2005). Usage of time-series analysis is common among these methods since the information obtained from a single message has been shown to be insufficient for predicting failures (Pinheiro et al., 2007; Yamanishi and Maruyama, 2005). Still it is an open question how to find the right pattern(s) (Fulp et al., 2008).

Salfner et al. (2006) present an approach called Similar Events Prediction (SEP) based on the recognition of suspicious patterns of error events (Table 1). SEP is shown to outperform the other failure prediction techniques and to achieve a precision of 80% and recall of 92%. Li et al. (2007) use the Cox proportional hazard model to provide a rigorous statistical prediction of system failure events. Frequent failure signatures are used as covariates. Vilalta and Ma (2002) first detect patterns in event sequences by using association rule mining techniques. Then patterns are combined into a rule-based model for prediction. The false negative error rate decreases significantly as the time window increases. Lee et al. (1991) analyze automatically generated event logs from fault tolerant systems. Multivariate statistical techniques (i.e., factor analysis and cluster analysis) are used to investigate error and failure dependency among different system components. Liang et al. (2007) predict failures of IBM's BlueGene/L from event logs containing reliability, availability and serviceability data. Temporal compression is used to include all events at a single location occurring with inter-event times lower than some threshold. Spatial compression serves to

Table 1
Overview of related work.

Reference	Approach	Results	Type of data	Limitations
Vilalta and Ma (2002)	Patterns are detected in event sequences by using association rule mining techniques. Then patterns are combined into a rule-based model for prediction	The false negative error rate decreases significantly as the time window increases	Artificial data and real data	The success of the algorithm is shown to be contingent on the existence of patterns preceding target events
Salfner et al. (2006)	Similar Events Prediction (SEP) based on the recognition of suspicious patterns of error events	SEP is shown to outperform the other failure prediction techniques and to achieve a precision of 80% and recall of 92%	Real data	Computational complexity of training grows heavily if the amount of training data increases
Pai and Hong (2006)	Simulated annealing algorithms are used to select the parameters of an SVM model to forecast software reliability	Experimental results show that the proposed model outperforms the other methods in predicting the inter-failure time	Real data	Advanced searching techniques for determining the suitable parameters could be used to increase the forecasting accuracy
Li et al. (2007)	Cox PH model is used to provide a rigorous statistical prediction of system failure events. Frequent failure signatures are used as covariates	The method is shown to handle effectively the situation of a long event sequence and a large number of event types in the sequence	Real data	Failure prediction for multiple event sequences should be inspected
Liang et al. (2007)	Temporal compression is used to include all events at a single location occurring with inter-event times lower than some threshold. Spatial compression serves to include all messages that refer to the same location within some time window. Four predictors are compared	With a prediction window size Δ of 12 h, 3 predictions perform reasonably well (F measure higher than 60%, precision higher than 50%, and recall higher than 70%)	Real data	The prediction window size Δ impacts on the prediction accuracy. As Δ becomes smaller, the prediction difficulty rapidly increases, leading to much degraded performances. Only the nearest neighbour predictor sustains a much slower degradation. Even with $\Delta = 1$ h, F measure is above 20%, and recall is 30%
Fulp et al. (2008)	SVMs are applied on data extracted from system log files to determine which sequence are precursors to failure	The spectrum-representation of messages combined with a SVM classifier is shown to achieve about 75% as true positive rate and about 25–30% as corresponding false positive rate. SVM using the spectrum-representation of messages is shown to outperform SVM using only aggregate features	Real data	The performance of the system could be improved by leveraging the message content, instead of solely relying on the tag value; in this case, adding more message information should be balanced with the sequence length, since the number of features grows exponentially. Moreover, more research is needed to study the impact of message diversity, which can be the result of machine purpose and log generation rates

include all messages that refer to the same location within some time window.

Support Vector Machines (SVMs) have been employed to solve non-linear regression and time series problems. SVMs have been also successfully applied to solve prediction problems when studying, e.g., time series (Tay and Cao, 2001; Cao, 2003), air quality (Wang et al., 2003), wind speed (Mohandes et al., 2004), forum hotspots (Li and Wu, 2010), and cancer (Chuang et al., 2011). In some cases SVMs have been applied to forecasting software reliability. Pai and Hong (2006) elucidate the feasibility of the use of SVMs to predict software reliability. Simulated annealing algorithms are used to select the parameters of a SVM model. The experimental results (in industrial settings) show that the proposed model outperforms the other methods in predicting the inter-failure time.

Aggregate features are often used for SVM-based classification (Liang et al., 2007; Xue et al., 2007; Yamanishi and Maruyama, 2005), e.g., the average number of messages during a period of time. However it is important to exploit the sequential nature of system messages. Fulp et al. (2008) apply SVMs on data extracted from log files to determine which sequences are precursors to failure. Experimental results show that the spectrum-representation of messages combined with a SVM classifier can achieve about 75% as true positive rate and about 25–30% as corresponding false positive rate. SVM using the spectrum-representation of messages is shown to outperform SVMs using only aggregate features. We compare our results with those achieved in the experiments in Fulp et al. (2008).

3. Background

3.1. Log files

Log files are important for managing computer systems since they provide a history or an audit trail of events. In this context an event is a change of a system status, such as a user login or an application failure.

Typically, system log files are text files that consist of messages sent to the logging service by applications. The applications can send information to the logging service process, which stores the messages in a text file in an arrival order. The logging service is primarily responsible for managing the log file while a content of the message is created by the application.

We use log files collected during approximately 3 months of work in a large European company that prefer to remain anonymous. Each log entry consists of seven fields. The Time field is the time when the message was recorded. One field stores the name of the running Application. Two fields identify the machine: Server and Computer. The logged-in user is reported in the UserName field, and Severity contains the level of severity of the message. There are three levels of severity: Information, Warning, and Error. The size on disk of the dataset is 0.8 GB.

The attributes of each dataset (i.e., log file) are listed in Table 2; Fig. 1 shows an example file. Due to the sensitive nature of the data presented here, data elements have been masked. In the example, the recorded messages are about 10 min of work of the user John, using the application "App1" on his computer "456" on server "123".

Table 2
Fields of the log messages in the analyzed dataset.

Field name	Description
Time	Time when a message was recorded
Application	Running application
Server	Machine sending a message
Computer	
UserName	Name of a logged user
Operation	Performed operation
Severity	Level of severity of a message

Fully automated and non-invasive data collection has been recognized as a successful approach (Coman et al., 2009). The full automation reduces the costs associated with data collection and ensures continuous and accurate measurements. The completely non-invasive collection allows developers to concentrate on their tasks as usual and thus it does not affect their efficiency. Log files can be automatically and non-invasively collected. Moreover, the approach presented in this paper needs just the following information to be available in the log files: the performed operation, the severity of the message, and the timestamp. Thus, this approach can be easily put into practice without much effort.

3.2. Random Indexing

In text analysis, Sahlgren (2005, 2006) proposes a word space model called RI as an alternative to Latent Semantic Analysis (LSA)-like models (Landauer et al., 1998) that first constructs a large co-occurrence matrix and then uses a separate dimension reduction phase. The main idea of RI is to accumulate context vectors based on occurrence of words in contexts. This technique is inherently incremental and does not require a separate dimension reduction phase.

The RI technique can be described as a two-step operation:

1. Each word is assigned a unique and randomly generated representation called *index vector*. The index vectors are sparse, high dimensional, and ternary, meaning that their dimensionality is on the order of thousands and they consist of a small number of randomly distributed +1 s and −1s, with the rest of the elements of the vectors set to 0. Each context is also assigned an initially empty context vector that has the same dimensionality as the index vector.
2. The text is scanned to find *context vectors*. Every time a word occurs in a context (e.g., within a sliding context window) the d-dimensional index vector of the context is added to the context vector for the word considered. Thus, words are represented by d-dimensional context vectors that are the sum of the words' contexts.

This methodology represents a radically different way of conceptualizing how context vectors are constructed. In the "traditional" view, firstly a co-occurrence matrix is constructed and then context vectors are extracted. In the RI approach, on the other hand, the context vectors are accumulated first. The co-occurrence

Time	Application	Server	Computer	UserName	Severity	Operation
01/05/11 10:50	App1	123	456	John	Information	Log in
01/05/11 10:52	App1	123	456	John	Information	Oper1
01/05/11 10:53	App1	123	456	John	Warning	Oper2
01/05/11 10:56	App1	123	456	John	Information	Oper3
01/05/11 10:57	App1	123	456	John	Warning	Oper4
01/05/11 11:00	App1	123	456	John	Error	Oper5

Fig. 1. Example of a log file in the dataset. Sensitive data elements have been masked.

matrix may be obtained by collecting the context vectors as rows of the matrix (Sahlgren, 2006).

For example, let us consider the sentence 'A friend in need is a friend indeed' (Chatterjee and Mohan, 2008). Let the dimension of the index vector be 10, and let the context be defined as one preceding and one succeeding word. Let 'friend' be assigned a random index vector: [0 0 0 1 0 0 0 0 −1 0] and 'need' be assigned a random index vector: [0 1 0 0 −1 0 0 0 0 0]. Then, to compute the context vector of 'in', RI sums up the index vectors of its context. Since the context is defined as one preceding and one succeeding word, the context vector of 'in' is the sum of index vectors of 'friend' and 'need' and is equal to [0 1 0 1 −1 0 0 0 −1 0] (Chatterjee and Mohan, 2008).

RI gives the following advantages (Raga and Raga, 2010; Sahlgren, 2005, 2006):

1. It is an incremental method, i.e., the context vectors can be used for similarity computations even with a small number of examples. By contrast, most other word space methods require the entire data to be sampled before similarity computations can be performed.
2. It uses fixed dimensionality, meaning that new data does not increase the dimensionality of the vectors. Increasing dimensionality can lead to significant scalability problems in other word space methods.
3. It uses implicit dimension reduction, since the fixed dimensionality is much lower than the number of words in the data. This leads to a significant gain in processing time and reduction of memory consumption.

Sahlgren and Cöster (2004) use RI to improve the performance of SVMs in text categorization. In this paper we use RI to (1) represent sequences of operations, characterizing each operation in terms of the context (i.e., operations within a sliding context window) in which it appears, and (2) improve the performance of the SVMs.

3.3. Support Vector Machines

SVMs (Vapnik, 1995) have been applied successfully in different fields, such as image retrieval, handwriting recognition, gene profiling, and text classification (Cortes, 1995; Joachims, 1998; Sahlgren and Cöster, 2004; Tong and Chang, 2001; Xu et al., 2010). SVM is one of the top 10 data mining algorithms identified by the IEEE International Conference on Data Mining (ICDM) in 2007 (Wu et al., 2007); it solves a mathematical optimization problem to find the separating hyperplane that has maximum margin between two classes. By maximizing the margin, the capacity or complexity of a function class (separating hyperplanes) is minimized.

Let $\{(\vec{x}_1, y_1), \ldots, (\vec{x}_l, y_l)\}$ be a set of training examples, where $\vec{x}_i \in R^n$, $y \in \{0, 1\}$. The SVM separates these examples by a hyperplane defined by a weight vector $\vec{w}$ and a threshold b. The weight vector $\vec{w}$ determines a direction perpendicular to the hyperplane, while b determines the distance to the hyperplane from the origin. A new example $\vec{z}$ is classified according to which side of the hyperplane it belongs. The hyperplane is uniquely defined by the *support vectors*.

When examples are not linearly separable, the SVM algorithm allows to use slack variables to allow classification errors and to map examples to a (high-dimensional) feature space. In this feature space, a separating hyperplane can be found such that, when mapped back to input space, it describes a non-linear decision function (Sahlgren and Cöster, 2004). The implicit mapping is performed by a kernel function that expresses the inner product between two examples in the desired feature space. In our experiments, we use the following three standard kernel functions:

- The basic linear kernel: $K(\vec{x}_i, \vec{z}) = x_i \cdot \vec{z}$.
- The polynomial kernel: $K(\vec{x}_i, \vec{z}) = (x_i \cdot \vec{z})^d$.
- The radial basis kernel: $K(\vec{x}_i, \vec{z}) = \exp(-\gamma||\vec{x}_i - \vec{z}||^2)$.

For all the experiments we select as default values $d = 3$ for the polynomial kernel and $\gamma = 1/\dim(data)$ for the radial basis (Akbani et al., 2004).

When facing with imbalanced datasets, where the number of negative instances far out numbers the positive instances, the performance of SVMs has been shown to drop significantly (Akbani et al., 2004). Application areas such as gene profiling, medical diagnosis, and credit card fraud detection have highly skewed datasets with a very small number of positive instances, which are hard to classify correctly but are important to detect nevertheless (Wu and Chang, 2003). With imbalanced data, the simplest hypothesis is often the one that classifies almost all instances as negative. There have been several alternative proposals for coping with skewed datasets, e.g., (1) to preprocess the data by under-sampling the majority class or oversampling the minority class to create a balanced dataset, or (2) to bias the classifier so that it gives more attention to the positive instances. One possible approach to do that is to increase the penalty associated with misclassifying the positive class relative to the negative class (Akbani et al., 2004; Japkowicz, 2005; Maloof, 2003; Osuna et al., 1997; Provost and Fawcett, 2001; Veropoulos et al., 1999).

Weighted SVMs implement cost-sensitive learning for SVM modeling. The basic idea is to assign a larger penalty value to false negatives than false positives (Akbani et al., 2004; Osuna et al., 1997; Veropoulos et al., 1999). Without loss of generality the cost for a false positive may be always 1; the cost for a false negative is usually suggested to be the ratio of negative samples over positive samples (Tang et al., 2009).

3.4. Performance assessment

Choosing the "best" candidate among different models involves performance assessment and detailed comparison. Many performance measures may be applied to perform this comparison.

A contingency table (sometimes called confusion matrix) is a convenient way to tabulate statistics for evaluating the quality of a model. In Table 3, TP, FP, TN, and FN stand for true positive, false positive, true negative, false negative counts, respectively. PP and PN stand for predicted positive/negative; and Pos and Neg stand for actual positive/negative.

In this paper we consider only metrics that can be defined in terms of counts in a contingency table; this excludes, e.g., the metrics that use model complexity (Flach, 2003; Jiang et al., 2008). The most relevant metrics of this type are reported in Table 4.

We use Receiver Operating Characteristics (ROC) space to analyze the performance of the classifier. False positive rate is plotted against the true positive rate. The graph (Fig. 2) shows the trade-off benefits (true positives) and costs (false positives). The points (0,0) and (1,1) represent the training-free classifiers *AlwaysNegative* and *AlwaysPositive*; the point (0,1) represents the ideal classifier, and (1,0) represents the classifier which gets it all wrong. The ascending diagonal (0,0)–(1,1) represents random training-free behavior:

Table 3
Contingency table (a.k.a., confusion matrix).

		Actual value	
		Pos	Neg
Prediction outcome	PP	TP	FP
Computer	PN	FN	TN

PP, predicted positive; PN, predicted negative; Pos, actual positive; Neg, actual negative; TP, true positive; FP, false positive; TN, true negative; FN, false negative.

Table 4
Derivations from a confusion matrix: the most relevant metrics for classification performance evaluation (Akbani et al., 2004; Menzies et al., 2007).

Name (a.k.a.)	Definition
True positive rate (TPR, sensitivity, recall)	$\frac{TP}{TP+FN}$
False positive rate (FPR, fall out)	$\frac{FP}{FP+TN}$
Accuracy	$\frac{TP+TN}{TP+TN+FP+FN}$
True negative rate (TNR, specificity)	$\frac{TN}{TN+FP}$
Balance	$1-\frac{\sqrt{(0-FPR)^2+(1-TPR)^2}}{\sqrt{2}}$

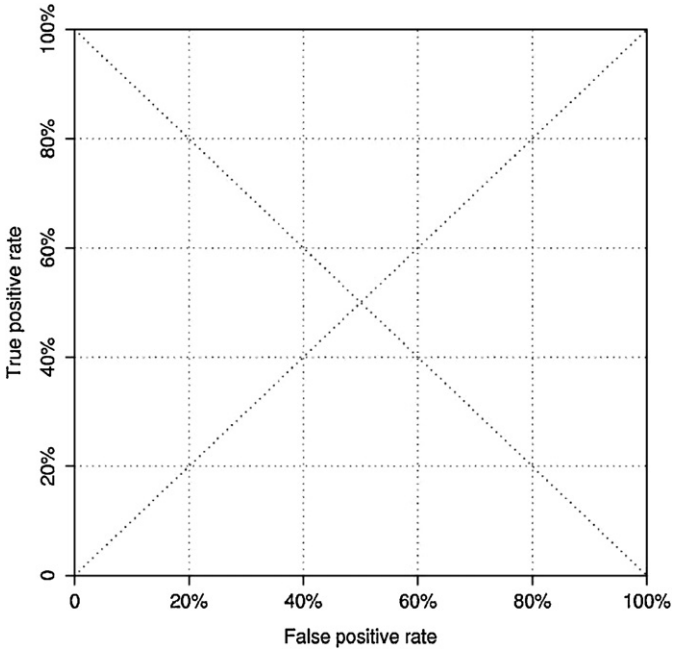

Fig. 2. ROC space structure, from Flach (2003).

any point (p,p) can be obtained by predicting positive with probability p and negative with probability $(1-p)$. The upper left triangle contains the classifiers that perform better than random, while the lower right triangle contains those performing worse than random. The descending diagonal (0,1)–(1,0) represents classifiers that perform equally well on both classes (TPR = 1 − FP); on the left of this line we find classifiers that perform better on the negatives than on the positives, on the right performance on the positives dominates (Flach, 2003).

Balance is defined in (Menzies et al., 2007) as the Euclidean distance from the sweet spot FPR = 0, TPR = 1 to a pair of (FPR; TPR). For convenience, we (1) normalize balance by the maximum possible distance across the ROC square ($\sqrt{2}$) and (2) subtract this from 1. This way, better and higher balances fall closer to the desired sweet spot of FPR = 0, TPR = 1. In particular, classifiers having balance higher than 0.5 fall in the upper left triangle of the ROC space. Thus, we consider 0.5 as a threshold for balance.

Ideally, we seek predictors that maximize accuracy, TPR, and TNR. Note that maximizing any of these does not imply high values for the others. Accuracy is a good measure of a learner's performance when the possible outcomes occur with similar frequencies, and it is not suggested when class distributions are uneven (Menzies et al., 2007). Therefore, this paper assesses its learned predictors using balance, TPR, FPR, and not accuracy.

3.5. Training and testing the model

The concept of splitting the data into two parts (not necessarily of equal size) to assess the performance of the model is by no means new. The available dataset D is divided into two disjoint subsets: the training set D_{train} to develop the model, and the test set (a.k.a., holdout set) D_{test} for testing the model to assess its performance. The historical background of this method is provided by Stone (1974, 1978) and Geisser (1975); useful discussions are given by Snee (1977) and Mosteller and Tukey (1977), who consider aspects of data splitting from a researcher's perspective (Picard and Cook, 1984).

The performance of classification methods is commonly evaluated by cross-validation (CV) (Allen, 1974; Stone, 1978; Wahba and Wold, 1975; Wold, 1978) or Monte Carlo CV (MCCV) (Picard and Cook, 1984; Shao, 1993; Xu et al., 2004). In m-fold CV, the n observations are divided into m (approximately) equally sized groups. In the kth iteration, the kth group is considered as test dataset, whereas the remaining $m-1$ groups form the training set which is used for classifier construction. This classifier is then used to predict the observations from group k. After the m iterations, the error rate is estimated as the proportion of misclassified observations. MCCV (a.k.a., subsampling or random splitting) also consists of several iterations in which the dataset is split into training and test sets. In contrast to CV, the test sets are not chosen to form a partition of the whole dataset but drawn randomly (without replacement) from the n observations at each iteration. The number of iterations is fixed by the user and can be as high as computationally feasible (usually higher than 200), leading to a more robust estimation than cross-validation (Smyth, 1996). The size ratio between training and test datasets is also fixed by the user. (Boulesteix, 2007). Normally, 60–80% of the total sample is used to estimate the model; the remaining 20–40% sample is set aside to validate the model. The larger D_{train}, the better is the classifier; the larger D_{test}, the better are its performances. In our experiments, we used MCCV with 1000 iterations; since we aimed at a good classifier with good performances, we chose to use 60% of the dataset as training set.

4. Approach

4.1. Structure of the approach

We propose a technique to predict the failure of a running software system using log files. The idea is to develop tools that read logs of a running system and signal the possible crash of this system.

Fig. 3 represents a schematic view of the proposed approach. While the system is running, log files are collected to track the actual execution path. In this work we look at the running system as a "black box", meaning that we do not have any other

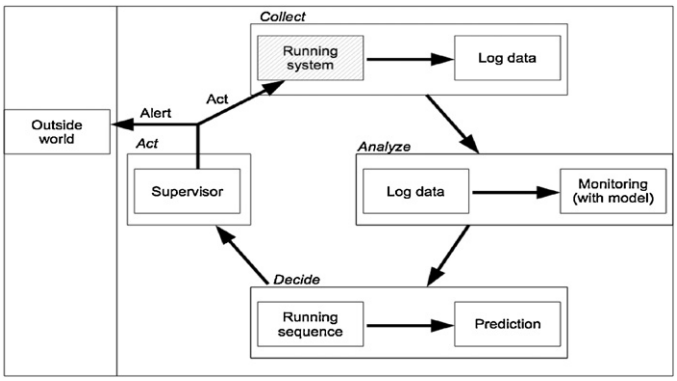

Fig. 3. Schema of the proposed approach.

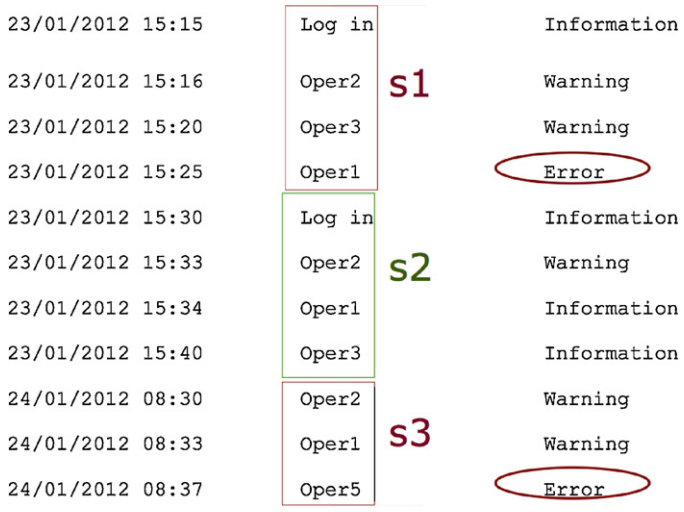

Fig. 4. Example of the process of extraction of sequences.

information about the system (e.g., the behavior in the memory could be used as an indicator of failure as well) except the log files.

The monitoring process takes log data as input and, based on the analysis performed (i.e., RI and SVM), gives the supervisor a message indicating the "likely failure" for the running system. The supervisor can act directly on the running system to avoid the predicted failure or send an alert to the outside world. Possible actions include: abort the running system, restart it, dynamically load components, or inform the running system if it is a suitably structured autonomic system (Müller et al., 2009).

4.2. Structure of the monitoring process

4.2.1. Dimensional reduction of the problem and data preparation

To get from raw log files to RI input, the following steps are performed:

1. Data are parsed to extract, for each event in the log file, the operation performed and the corresponding timestamp and severity.
2. Duplicate rows are deleted, together with logs where the operation, the timestamp, or the severity is missing.
3. Sequences of operations are extracted; a new sequence starts either if there is a 'Log in' operation or if a day changes. A sequence is labeled as "failure" if it contains at least one severity "Error". An example is shown in Fig. 4. Sequence 1 and sequence 2 (i.e., s1 and s2) start with a 'Log in'; s3 starts because the day is changed. In s1 and in s3, one operation has severity 'Error'; thus, s1 and s3 are labeled as 'failure'.
4. A text file is created as input for RI, where each line represents a sequence of operations (i.e., a sentence). As an example, Fig. 5 shows the input file corresponding to the example in Fig. 4.

4.2.2. SVM training and analysis of the results

Following the guidelines in (Sahlgren and Cöster, 2004), in each application the monitoring process acts as follows:

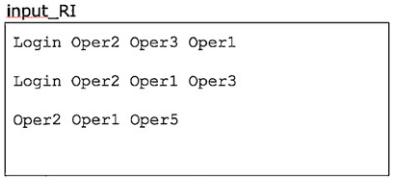

Fig. 5. An example of input file for RI (corresponding to the log file in Fig. 4).

1. It assigns index vectors to operation types.
2. It scans all the sequences of operations and finds context vectors for each word (i.e., operation) using RI (Jurgens and Stevens, 2010) where the length of semantic vectors is 1000 and window size (i.e., number of operations to consider in each direction) is 1. In our preliminary experiments, increasing the window size did not improve the results appreciably. Thus, we used the smallest context to deal with short sequences as well.
3. It uses context vectors to generate representations for sequences of operations. This is done for every sequence by summing the (weighted) context vectors of the operations that occur in the particular sequence. Note that summed vectors result in tf-weighting,[1] since a word's vector is added to the text's vector as many times as the word occurs in the text (Sahlgren and Cöster, 2004).
4. Vectors representing sequences are used to train a SVM. The current sequence of operations of the running system is then classified according to the obtained model.

5. Experiments and results

5.1. Goal and data

We have applied our approach to industrial data to select the best performing type of SVM (i.e., weighted or not) and its kernel. Moreover, we aimed at evaluating objectively our prediction system applying the most suggested performance evaluation techniques (Flach, 2003; Jiang et al., 2008). We used log files collected in a large European company that prefer to remain anonymous. The study spanned a period of approximately 3 months.

Initially, the team was composed of 15 developers, but 2 more joined the team at the beginning of the second month of our study. The developers are all between 30 and 40 years old. They all hold university degrees in computer-related areas and have from 10 to 15 years of programming experience. The team works on several projects, mainly in C#. They are an Agile team, using a customized version of Extreme Programming (XP). In particular, they use weekly iterations, pair programming, user stories, and the test-first approach. They use pair programming on an "as needed" basis. The team has been using XP for more than two years previously to our study. The working space of the team consists in an open space, where each member has her/his own personal space. Therefore, informal communication between the developers is easily possible.

Table 5 describes the six applications in the dataset. For privacy reasons, we could not access the code to extract metrics and provide a more exhaustive description of the applications.

After the preprocessing phase, sequences were sampled 1000 times using Monte Carlo methods to obtain calibration sets (60%) and validation sets (40%).

We applied SVM with linear, polynomial, and radial basis kernel. The cost of misclassifying points was 100 in each case to force the creation of a more accurate model (Tang et al., 2009).

To cope with skewed datasets weighted SVMs were also applied; the cost for a false positive was always 1, while the cost for a false negative was the ratio of negative samples over positive samples (Tang et al., 2009).

[1] Tf-weighting is the weighting scheme referred to as *term frequency*, usually denoted $tf_{t,d}$, with the subscripts representing the term t and the document d. A weight is assigned to each term in a document; the weight is defined as the number of occurrences of term t in document d (Manning et al., 2008).

Table 5
Description of the applications in the dataset.

App.	Type	Maturity	Evolution	Complexity	Size	Number of sequences	Pos[a] (%)
A1	Custom ERP	Medium	Constantly modified	Medium	Large	12,765	0.84
A2	Management of historical data on mechanic components	High	Mildly modified	Medium	Medium	718	15.88
A3	Signals about variations in the ERP system	High	Rarely modified	Low	Medium	60	23.33
A4	Calibration and interface with proprietary hardware	High	Rarely modified; going to be dismissed	Low	Small	343	12.83
A5	Radio communication	High	Constantly modified	Extremely high	Very large	713	4.77
A6	Mechanical simulations	High	Constantly modified	Extremely high	Very large[b]	8593	1.23

[a] Pos = failures (percentage over *n*).
[b] The biggest and most critical application in the company.

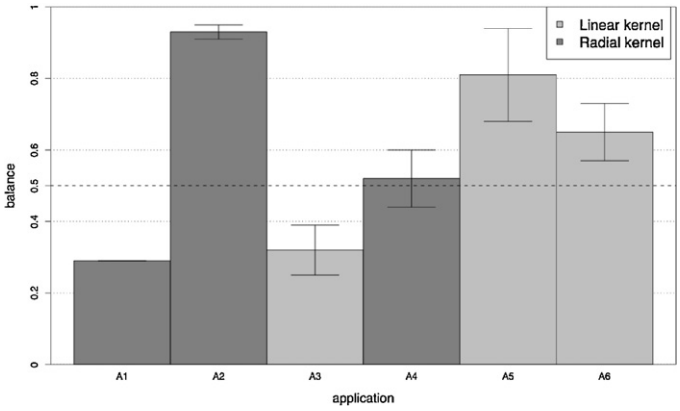

Fig. 6. Balance (with standard deviation) of the best performing SVMs.

5.2. Results

For each application, the best performing SVM was selected based on TNR and balance. Table 6 shows the results of this analysis. TNR is higher than 0.96 in all the applications.

Fig. 6 shows that balance is higher than 0.50 in four out of six cases, as four bars go over the dashed line representing the threshold for balance.

Fig. 7 shows the ROC space of the best performing SVMs listed in Table 6. The mean values of TPR and FPR of the 1000 experiments are plotted together with their standard deviation. Five out of six SVMs fall in the upper left triangle; therefore, they perform better than random. In particular, these five points are left of the descending diagonal (0,1)–(1,0); thus, these five classifiers perform better on the negatives than on the positives. Classification in A1 is *AlwaysNegative*, since the corresponding point is (0,0).

As shown in Table 6, SVMs had almost perfect TNR but poor TPR. TNR may be so high because data are imbalanced; thus, SVMs classify almost all instances as negative. Any algorithm that tries to improve on it inevitably sacrifices some specificity to improve TPR. There have been several proposals for coping with skewed datasets

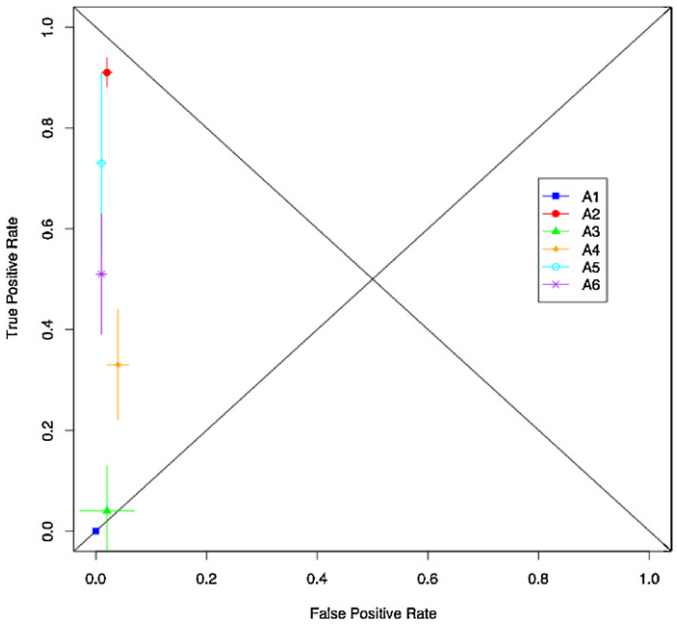

Fig. 7. ROC space of the best performing SVMs.

(Akbani et al., 2004; Japkowicz, 2005; Maloof, 2003; Provost and Fawcett, 2001; Veropoulos et al., 1999). We address this issue applying weighted SVMs, which implement cost-sensitive learning (Akbani et al., 2004; Osuna et al., 1997; Tang et al., 2009; Veropoulos et al., 1999).

According to results, weighted SVMs sacrifice some specificity (i.e., TNR) to improve the TPR; anyway, TNR still remains higher than 0.80 in four out of six operations, as shown in Table 7.

Fig. 8 shows that weighted SVMs balance is higher than 0.50 in four out six cases, as it was for SVMs (Fig. 6). Fig. 9 shows the comparison between the balance of SVMs and of weighted SVMs.

Fig. 10 shows the ROC space of the best performing weighted SVMs listed in Table 7. Five out of six points are in the upper left

Table 6
SVMs performance (best performing kernels).

App	SVM performance				
	Kernel	TNR (std)	TPR (std)	FPR (std)	Balance (std)
A1	Radial	1.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.29 (0.00)
A2	Radial	0.98 (0.01)	0.91 (0.03)	0.02 (0.01)	0.93 (0.02)
A3	Linear	0.98 (0.05)	0.04 (0.09)	0.02 (0.05)	0.32 (0.07)
A4	Radial	0.96 (0.02)	0.33 (0.11)	0.04 (0.02)	0.52 (0.08)
A5	Linear	0.99 (0.01)	0.73 (0.18)	0.01 (0.01)	0.81 (0.13)
A6	Linear	0.99 (0.01)	0.51 (0.12)	0.01 (0.01)	0.65 (0.08)

Table 7
Weighted SVMs performance (best performing kernels).

App	SVM performance				
	Kernel	TNR (std)	TPR (std)	FPR (std)	Balance (std)
A1	Radial	0.51 (0.11)	0.50 (0.13)	0.48 (0.11)	0.48 (0.03)
A2	Radial	0.98 (0.01)	0.93 (0.04)	0.02 (0.01)	0.94 (0.02)
A3	Linear	0.80 (0.15)	0.17 (0.18)	0.20 (0.15)	0.36 (0.09)
A4	Radial	0.84 (0.04)	0.76 (0.11)	0.16 (0.04)	0.79 (0.06)
A5	Linear	0.95 (0.02)	0.85 (0.09)	0.05 (0.02)	0.88 (0.06)
A6	Linear	0.95 (0.04)	0.72 (0.16)	0.05 (0.04)	0.80 (0.18)

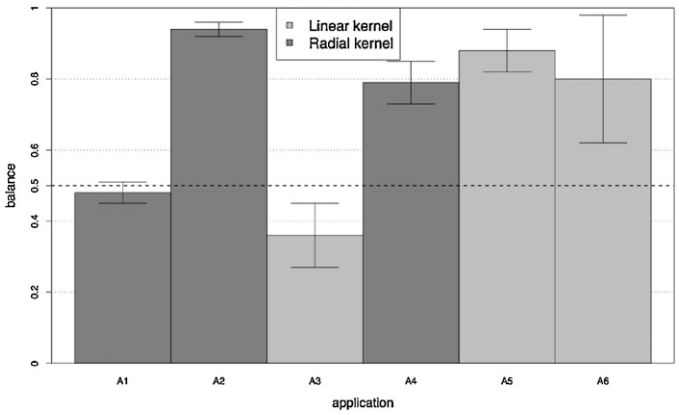

Fig. 8. Balance (with standard deviation) of the best performing weighted SVMs.

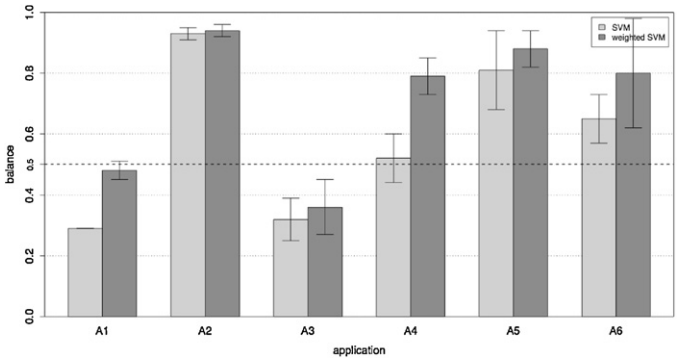

Fig. 9. Balance of SVMs and weighted SVMs: a comparison.

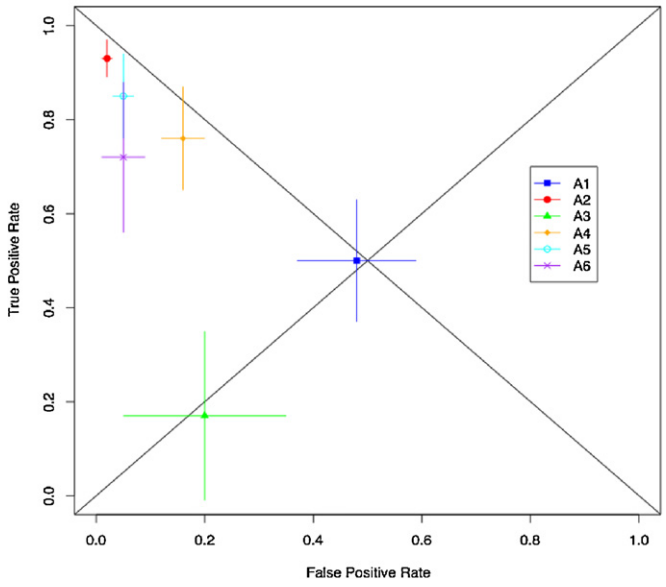

Fig. 10. ROC space of the best performing weighted SVMs.

triangle; therefore, they perform better than random. It emerges from the comparison of this plot to Fig. 7 that A4 and A1 have improved their performance. In particular, A1 has now left the (0,0) point. Our approach outperforms in accuracy the method proposed in (Fulp et al., 2008), where authors use SVMs to predict computer failure events using sub-sequences from system log.

Table 8
Summary of the results obtained in this study.

	SVMs	Weighted SVMs
TNR	Always higher than 0.96	Always higher than 0.80
TPR	Lower than 0.50 in 4 out of 6 cases	Lower than 0.50 in 1 out of 6 cases
Balance	Higher than 0.50 in 4 out of 6 cases	Higher than 0.50 in 4 out of 6 cases
		Always higher than non weighted SVMs results

6. Conclusions and limitations

In this paper we introduce a new approach for predicting failures of a running system using log files; we use RI to represent sequences of operations extracted from log files, then we apply SVMs to classify them as either fail or non-fail. This approach allows us to exploit the sequential nature of system messages.

As expected, we found that SVMs were almost perfect in classifying non-failures correctly but poor in identifying failures. To solve this issue we applied weighted SVMs to implement cost-sensitive learning, assigning a larger penalty value to false negatives than false positives. According to results (Table 8), some specificity (i.e., TNR) was sacrificed to improve sensitivity (i.e., TPR). Anyway, TNR still remains higher than 0.80 in four out of six applications. Overall, weighted SVMs are very reliable in predicting both failures and non-failures of the system.

Table 9 reports the possible threats of validity of our study according to the list reported in Wohlin et al. (2000).

6.1. Generalizability

We applied the most suggested techniques (Flach, 2003; Jiang et al., 2008) to evaluate objectively the performance of our approach. Nevertheless, misclassification carries the risk of system failure also associated with cost implications. Thus, when applying the approach in another context, the selection of the "best" model should consider project cost characteristics, which are specific in each development environment (Jiang et al., 2008).

The log files need to contain the performed operation, the severity of the message, and the timestamp. Therefore, this approach can be easily put into practice in different contexts. Moreover, we have shown that our approach can be applied in different conditions, since we have considered a set of application heterogeneous in terms of size, complexity, type and maturity (Table 5).

7. Future work

Our results contribute to the research on failure prediction in exploiting the sequential nature of log messages while classifying sequences as either fail or non-fail, thus allowing to abort the running system, or restart it, or take corrective actions to avoid the failure. The proposed model could be particularly useful dealing with autonomic systems. Autonomic systems could be instructed to receive signals of likely failures and upon reception of such signals could start a suitable recovery procedure (Müller et al., 2009).

Given the results of this approach, the company where this study took place has agreed to extend the study for an additional period of one year. After the 3-months study presented in this work, we collected feedback from the company. We are now performing experiments to address the raised issues. In particular, we want to understand if our results can be improved by changing the definition of context, i.e., the dimension of the sliding window in the RI algorithm. Another aspect that we will evaluate is the possibility of predicting the occurrence of a failure using only an initial portion of a sequence, so that there could be an early estimation of failure,

Table 9
Threats to validity of our experiment.

Conclusion validity	
Random heterogeneity of subjects	The applications are quite heterogeneous, thus the variation in the performances might be due to individual differences. Anyway, our experiments seem to exclude the influence on performance of the number of sequences and of the percentage of failures
Internal validity	
History	The performance of our approach could change over time, since the characteristics of the applications may evolve (e.g., the prediction may be more accurate with a higher percentage of failures). Analyzing the performance over time may solve this possible threat
Maturation	
External validity	
Interaction of selection and treatment	We applied our approach to industrial data and analyzed six different applications. Still, drawing general conclusions from empirical studies in software engineering is complicated because any process depends to a large degree on a potentially large number of relevant context variables. Thus, the replication of the analysis on more industrial datasets is needed to generalize beyond the specific environment in which it was conducted

providing additional time to take corrective actions. Moreover, we will analyze how our approach influences positively the percentage of failures. Finally, we will investigate how we could consider other "black-box" properties or applications to predict failures; candidate properties include memory usage, number of open files, processor usage, etc. Our future goal is to study more in-depth our promising model to determine if we can generalize our results. To this end, we plan to replicate the analysis on more industrial datasets.

References

Adiga, N., Almasi, G., Almasi, G.S., Aridor, Y., Barik, R., Beece, D., Bellofatto, R., Bhanot, G., Bickford, R., Blumrich, M., Bright, A.A., Brunheroto, J., Cacaval, C., Castaños, J., Chan, W., Ceze, L., Coteus, P., Chatterjee, S., Chen, D., Chiu, G., Cipolla, T.M., Crumley, P., Desai, K.M., Deutsch, A., Domany, T., Dombrowa, M.B., Donath, W., Eleftheriou, M., Erway, C., Esch, J., Fitch, B., Gagliano, J., Gara, A., Garg, R., Germain, R., Giampapa, M.E., Gopalsamy, B., Gunnels, J., Gupta, M., Gustavson, F., Hall, S., Haring, R.A., Heidel, D., Heidelberger, P., Herger, L.M., Hoenicke, D., Jackson, R.D., Jamal-Eddine, T., Kopcsay, G.V., Krevat, E., Kurhekar, M.P., Lanzetta, A.P., Lieber, D., Liu, L.-K., Lu, M., Mendell, M., Misra, A., Moatti, Y., Mok, L., Moreira, J.E., Nathanson, B.J., Newton, M., Ohmacht, M., Oliner, A., Pandit, V., Pudota, R.B., Rand, R., Regan, R., Rubin, B., Ruehli, A., Rus, S., Sahoo, R.K., Sanomiya, A., Schenfeld, E., Sharma, M., Shmueli, E., Singh, S., Song, P., Srinivasan, V., Steinmacher-Burow, B.D., Strauss, K., Surovic, C., Swetz, R., Takken, T., Tremaine, R.B., Tsao, M., Umamaheshwaran, A.R., Verma, P., Vranas, P., Ward, T.J.C., Wazlowski, P., 2002. An Overview of the bluegene/l supercomputer. In: Proceedings of Supercomputing, p. 60.

Akbani, R., Kwek, S., Japkowicz, N., 2004. Applying support vector machines to imbalanced datasets. Machine Learning 3201 (7), 39–50.

Allen, D.M., 1974. The relationship between variable and data augmentation and a method of prediction. Technometrics 16, 125–127.

Boulesteix, A.-L., 2007. WilcoxCV: an R package for fast variable selection in cross-validation. Bioinformatics 23, 1702–1704.

Cao, L., 2003. Support vector machines experts for time series forecasting. Neurocomputing 51, 321–339.

Chatterjee, N., Mohan, S., 2008. Discovering word senses from text using Random Indexing. In: Proceedings of the 9th International Conference on Intelligent Text Processing and Computational Linguistics, Haifa, Israel, February 17–23, 2008, pp. 299–310.

Chuang, L.Y., Wu, K.C., Chang, H.W., Yang, C.H., 2011. Support Vector Machine-based prediction for oral cancer using four SNPs in DNA repair genes. Lecture Notes in Engineering and Computer Science 2188 (1), 426–429.

Coman, I.D., Sillitti, A., Succi, G., 2009. A case-study on using an Automated In-process Software Engineering Measurement and analysis system in an industrial environment. In: Proceedings of the International Conference on Software Engineering, Canada, May 16–24, 2009, pp. 89–99.

Cortes, C., 1995. Prediction of generalisation ability in learning machines. PhD Thesis, Department of Computer Science, University of Rochester.

Flach, P.A., 2003. The geometry of ROC space: understanding machine learning metrics through ROC isometrics. In: Proceedings of the 20th International Conference on Machine Learning, pp. 194–201.

Fu, S., Xu, C.Z., 2007a. Quantifying temporal and spatial fault event correlation for proactive failure management. In: Proceedings of Symposium on Reliable and Distributed Systems.

Fu, S., Xu, C.Z., 2007b. Exploring event correlation for failure prediction in coalitions of clusters. In: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis, Reno, NV, USA, November 15–21, 2007.

Fulp, E.W., Fink, G.A., Haack, J.N., 2008. Predicting computer system failures using support vector machines. In: Proceedings of the 1st Conference on Analysis of System Logs.

Geisser, S., 1975. The predictive sample reuse method with applications. Journal of the American Statistical Association 70, 320–328.

Gross, K.C., Bhardwaj, V., Bickford, R., 2002. Proactive detection of software aging mechanisms in performance critical computers. In: Proceedings of the 27th Annual NASA Goddard Software Engineering Workshop.

Japkowicz, N., 2005. Learning from imbalanced data sets: a comparison of various strategies. In: Proceedings of Learning from Imbalanced Data Sets Workshop.

Jiang, Y., Cukic, B., Ma, Y., 2008. Techniques for evaluating fault prediction models. Empirical Software Engineering 13 (5), 561–595.

Joachims, T., 1998. Text categorization with SVM: learning with many relevant features. In: Proceedings of the 10th European Conference on Machine Learning.

Jurgens, D., Stevens, K., 2010. The S-space package: an open source package for word space models. In: System Papers of the Association of Computational Linguistics.

Lan, Z., Zheng, Z., Li, Y., 2010. Toward automated anomaly identification in large-scale systems. In: Proceedings of Transactions on Parallel and Distributed Systems, pp. 174–187.

Landauer, T.K., Foltz, P.W., Laham, D., 1998. Introduction to latent semantic analysis. Discourse Processes 25, 259–284.

Lee, I., Iyer, R.K., Tang, D., 1991. Error/failure analysis using event logs from fault tolerant systems. In: Proceedings of the 21st International Symposium on Fault-Tolerant Computing.

Li, N., Wu, D.D., 2010. Using text mining and sentiment analysis for online forums hotspot detection and forecast. Decision Support Systems 48 (2), 354–368.

Li, Z., Zhou, S., Choubey, S., Sievenpiper, C., 2007. Failure event prediction using the Cox proportional hazard model driven by frequent failure sequences. IEE Transactions 39 (3), 303–315.

Liang, Y., Zhang, Y., Xiong, H., Sahoo, R., 2007. Failure prediction in IBM bluegene/l event logs. In: Proceedings of the International Conference on Data Mining.

Maloof, M.A., 2003. Learning when data sets are imbalanced and when costs are unequal and unknown. In: Proceedings of the 2nd Workshop on Learning from Imbalanced Data Sets.

Mannila, H., Toinoven, H., Verkamo, A.I., 1997. Discovery of frequent episodes in event sequences. Data Mining and Knowledge Discovery 1, 259–289.

Manning, C.D., Raghavan, P., Schütze, H., 2008. Introduction to Information Retrieval. Cambridge University Press.

Menzies, T., Greenwald, J., Frank, A., 2007. Data mining static code attributes to learn defect predictors. IEEE Transactions on Software Engineering 33 (1), 2–13.

Mohandes, M.A., Halawani, T.O., Rehman, S., Hussain, A.A., 2004. Support vector machines for wind speed prediction. Renewable Energy 29, 939–947.

Mosteller, F., Tukey, J.W., 1977. Data Analysis and Regression. Addison Wesley, Reading, MA.

Müller, H.A., Kienle, H.M., Stege, U., 2009. Autonomic computing: now you see it, now you don't—design and evolution of autonomic software systems. In: De Lucia, A., Ferrucci, F. (Eds.), Software Engineering International Summer School Lectures: University of Salerno, LNCS 5413. Springer-Verlag, pp. 32–54.

Osuna, E., Freund, R., Girosi, F., 1997. Support vector machines: training and applications. Technical Report, Massachusetts Institute of Technology, Cambridge, MA, USA.

Pai, P.F., Hong, W.C., 2006. Software reliability forecasting by support vector machines with simulated annealing algorithms. Journal of Systems and Software 79 (6), 747–755.

Pandalai, D.N., Holloway, L.E., 2000. Template languages for fault monitoring of timed discrete event processes. IEEE Transactions on Automatic Control 45 (5), 868–882.

Picard, R.R., Cook, R.D., 1984. Cross-validation of regression models. Journal of the American Statistical Association 79 (387), 575–583.

Pinheiro, E., Weber, W.D., Barroso, L.A., 2007. Failure trends in a large disk drive population. In: Proceedings of the Conference on File and Storage Technologies, pp. 17–29.

Provost, F.J., Fawcett, T., 2001. Robust classification for imprecise environments. Machine Learning 42 (3), 203–231.

Raga, R., Raga, J., 2010. Combining Random Indexing and instance-based learning to analyze sentential units of text. In: Proceedings of the 10th Philippine Computing Science Congress.

Sahlgren, M., Cöster, R., 2004. Using bag-of-concepts to improve the performance of support vector machines in text categorization. In: Proceedings of the 20th International conference on Computational Linguistics, Geneva, Switzerland, August 23–27, 2004.
Sahlgren, M., 2005. An introduction to Random Indexing. In: Proceedings of the 7th International Conference on Terminology and Knowledge Engineering.
Sahlgren, M., 2006. The Word-Space Model: using distributional analysis to represent syntagmatic and paradigmatic relations between words in high-dimensional vector spaces. PhD Thesis, Stockholm University.
Salfner, F., Schieschke, M., Malek, M., 2006. Predicting failures of computer systems: a case study for a telecommunication system. In: Proceedings of the 20th International Conference On Parallel and Distributed Processing Symposium, Rhodes Island, Greece, April 25–29, 2006.
Salfner, F., 2008. Event-based failure prediction: an extended hidden Markov model approach. Dissertation, Verlag, Berlin, Germany.
Sampath, M., Sengupta, R., Lafortune, S., 1994. Diagnosability of discrete event systems. In: Proceeding of the 11th International Conference on Analysis and Optimization of Systems Discrete Event Systems, Sophia, Antipolis, June 15–17, 1994.
Shao, J., 1993. Linear model selection by cross validation. Journal of the American Statistical Association 88, 486–494.
Srinivasan, V.S., Jafari, M.A., 1993. Fault detection/monitoring using time petri nets. IEEE Transactions on System, Man and Cybernetics 23 (4), 1155–1162.
Smyth, P., 1996. Clustering using Monte Carlo cross-validation. In: Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining, Portland, Oregon, 1996.
Snee, R.D., 1977. Validation of regression models: methods and examples. Technometrics 19, 415–428.
Stearley, J., Oliner, A.J., 2008. Bad words: finding faults in Spirit's syslogs. In: Proceedings of the International Symposium on Cluster Computing and the Grid, Lyon, France, May 19–22, 2008.
Stone, M., 1974. Cross-validatory and assessment of statistical predictions. Journal of the Royal Statistical, Series B 36, 111–133.
Stone, M., 1978. Cross-validation: a review. Mathematische Operationsforschung und Statistik 9, 127–139.
Tang, Y., Zhang, Y.Q., Chawla, N.V., Krasser, S., 2009. SVMs modeling for highly imbalanced classification. Systems, man, and cybernetics. Transactions on Cybernetics 39 (1), 281–288.
Tay, F.E.H., Cao, L., 2001. Application of support vector machines in financial time series forecasting. Omega: The International Journal of Management Science 29, 309–317.
Tong, S., Chang, E., 2001. Support Vector Machine active learning for image retrieval. In: Proceedings of the International Conference on Multimedia, pp. 107–118.
Vapnik, V., 1995. The Nature of Statistical Learning Theory. Springer-Verlag.
Veropoulos, K., Campbell, C., Cristianini, N., 1999. Controlling the sensitivity of support vector machines. In: Proceedings of Workshop Support Vector Machines.
Vilalta, R., Ma, S., 2002. Predicting rare events in temporal domains. In: Proceedings of the International Conference on Data Mining.
Wang, W., Xu, Z., Lu, J.W., 2003. Three improved neural network models for air quality forecasting. Engineering Computations 20, 192–210.
Wahba, G., Wold, S., 1975. A completely Automatic French Curve. Communications in Statistics 4, 1–17.
Wohlin, C., Runeson, P., Höst, M., Ohlsson, M.C., Regnell, B., Wesslén, A., 2000. Experimentation Software Engineering – An Introduction. Kluwer Academic Publishers.
Wold, S., 1978. Cross-validatory estimation of the number of components in factor and principal components models. Technometrics 20, 397–405.
Wu, G., Chang, E., 2003. Class-Boundary Alignment for Imbalanced Dataset Learning. ICM, Washington, DC.
Wu, X., Kumar, V., Ross Quinlan, J., Ghosh, J., Yang, Q., Motoda, H., McLachlan, G.J., et al., 2007. Top 10 algorithms in data mining. Knowledge and Information Systems 14 (1), 1–37.
Xu, H., Lemischka, I.R., Ma'ayan, A., 2010. SVM classifier to predict genes important for self-renewal and pluripotency of mouse embryonic stem cells. BMC Systems Biology 4 (173).
Xu, Q.S., Liang, Y.Z., Du, Y.P., 2004. Monte Carlo cross-validation for selecting a model and estimating the prediction error in multivariate calibration. Journal of Chemometrics 18, 112–120.
Xue, Z., Dong, X., Ma, S., Dong, W., 2007. A survey on failure prediction of large-scale server clusters. In: Proceedings of the International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, pp. 733–738.
Yamanishi, K., Maruyama, Y., 2005. Dynamic syslog mining for network failure monitoring. In: Proceedings of the International Conference on Knowledge Discovery in Data Mining, pp. 499–508.
Zheng, Z., Lan, Z., Park, B.-H., Geist, A., 2009. System log pre-processing to improve failure prediction. In: Proceedings of DSN'09.

Ilenia Fronza holds a PhD in Computer Science received from the University of Bolzano-Bozen, Italy, in 2012. She is currently a research fellow on a fixed term contract at the Free University of Bolzano-Bozen, Italy. Her current research interest focuses on data mining and computational intelligence, software process measurement and improvement, failure prediction.

Alberto Sillitti, PhD, PEng is Associate Professor at the Faulty of Computer Science of the Free University of Bolzano-Bozen, Italy. He holds a PhD in Electrical and Computer Engineering received from the University of Genoa (Italy) in 2005. He has been involved in several EU funded projects related to Open Source Software, Services Architectures, and Agile Methods in which he applies non-invasive measurement approaches. He has served as member of the program committee of several international conferences and as program chair of OSS 2007, XP 2010, and XP2011. His research areas include open source development, agile methods, software engineering, non-invasive measurement, mobile and web services. He is author of more than 80 papers published in international conferences and journals.

Giancarlo Succi is Professor with Tenure at the Free University of Bolzano-Bozen, Italy, where he directs the Centre for Applied Software Engineering. Before joining the Free University of Bolzano-Bozen, he has been Professor with Tenure at the University of Alberta, Edmonton, Alberta, Associate Professor at the University of Calgary, Alberta, and Assistant Professor at the University of Trento, Italy. The research interest of his involve multiple areas of software engineering, including open source development, agile methodologies, experimental software engineering, software engineering over the Internet, and software product lines and software reuse. He is a Fulbright Scholar.

Mikko Terho works in the Devices R&D Technology, Strategy and Architecture unit. His current interests are open source development around Qt, machine learning and semantic data manipulation. He also manages a team that develops prototypes for pervasive communication devices with novel Internet services. As a Nokia Fellow, he advices other Nokia R&D Groups in the area of system design, software architecture and component selection. He joined Nokia in 1983 and has since served in various research and development and managerial positions within the company. He is also a founding Director of Wireless Application Protocol (WAP) Forum Ltd. and founding Director of Symbian Ltd.

Jelena Vlasenko received her Bachelor degree in Computer Science from the Free University of Bolzano-Bozen, Italy, in 2010. She is currently a Master student at the Free University of Bolzano-Bozen, Italy, where she works on her thesis at the Center for Applied Software Engineering. Her current research interest focuses on agile methodologies, failure prediction, non-invasive measurement, software process measurement and improvement.