Unsupervised Machine Learning Techniques for Anomaly Detection with Multi-Source Data

Filippo Pacinelli

October 10, 2022

Anomalies

Anomaly Detection allows to:

- improve time performances
- improve process' efficiency
- save money
- ...

As such, it's a key problem in Research Centers, but also in the industrial sector, construction etc.

Problem

This work deals with anomaly detection in the Data Center of INFN-CNAF Institute.

Main challenges:

- different types of data (textual and numerical)
- completely unsupervised task
- thousands of machines to analyze, therefore some automatic mechanism was necessary

Data Available:

- **log data**: log messages of softwares running on the machines of the data center
- **monitoring data**: numerical sequences representing metrics to check the health status of machines.

Log Data

They are messages produced by logging systems. They represent messages related to the sucessful running of processes, but also failures, dangers or warnings.

index	timestamp	process_name	msg
0	1591447373.0	smartd	Device: /dev/sdb [SAT], CHECK POWER STATUS spins up disk (0x81 -> 0xff)
1	1591727610.0	auditd	Audit daemon rotating log files
2	1592093573.0	smartd	Device: /dev/sda [SAT], CHECK POWER STATUS spins up disk (0x81 -> 0xff)
3	1592404973.0	smartd	Device: /dev/sda [SAT], CHECK POWER STATUS spins up disk (0x81 -> 0xff)
4	1592440973.0	smartd	Device: /dev/sdb [SAT], CHECK POWER STATUS spins up disk (0x81 -> 0xff)
5	1592568773.0	smartd	Device: /dev/sdb [SAT], CHECK POWER STATUS spins up disk (0x81 -> 0xff)
6	1592619174.0	smartd	Device: /dev/sdb [SAT], CHECK POWER STATUS spins up disk (0x81 -> 0xff)
7	1592629973.0	smartd	Device: /dev/sdb [SAT], CHECK POWER STATUS spins up disk (0x81 -> 0xff)
8	1592640773.0	smartd	Device: /dev/sdb [SAT], CHECK POWER STATUS spins up disk (0x81 -> 0xff)
9	1592655608.0	auditd	Audit daemon rotating log files
10	1592677209.0	auditd	Audit daemon rotating log files
11	1592685778.0	smartd	Device: /dev/sdb [SAT], CHECK POWER STATUS spins up disk (0x81 -> 0xff)
12	1592775773.0	smartd	Device: /dev/sdb [SAT], CHECK POWER STATUS spins up disk (0x81 -> 0xff)
13	1592786412.0	auditd	Audit daemon rotating log files
14	1592808416.0	auditd	Audit daemon rotating log files
15	1592873380.0	auditd	Audit daemon rotating log files
16	1592908973.0	smartd	Device: /dev/sda [SAT], CHECK POWER STATUS spins up disk (0x81 -> 0xff)
17	1592982773.0	smartd	Device: /dev/sda [SAT], CHECK POWER STATUS spins up disk (0x81 -> 0xff)
18	1593001500.0	auditd	Audit daemon rotating log files
19	1593009773.0	smartd	Device: /dev/sdb [SAT], CHECK POWER STATUS spins up disk (0x81 -> 0xff)
20	1593022717.0	auditd	Audit daemon rotating log files

Figure: Example of occurrence of log messages on the machine cn-608-01-01.

Monitoring Data

They are registered by control systems and represent the relevant metric to understand the health status of a machine. They are registered with regular frequency (usually 5 minutes) and represent meaningful values like:

- the overall workload a machine
- the amount of CPU memory currently used
- the number of processes running simultaneously on a machine

Process Adopted

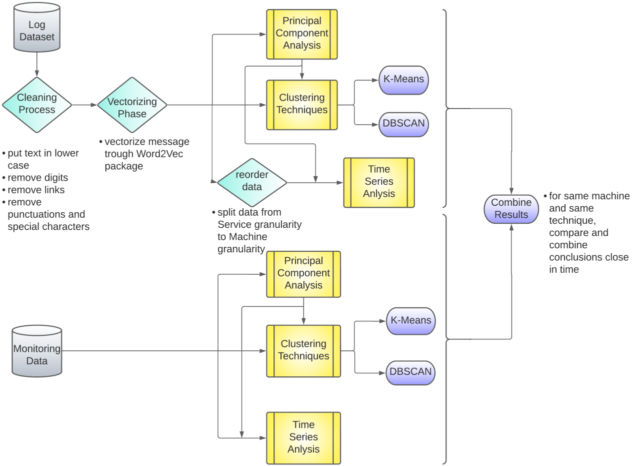


Figure: Scheme of the process adopted

Word2Vec

Word2Vec is a pre-saved model on Python's Gensim library which allows vectorization of texts.

- it's a 2-Layer Neural Network which is fed with hot-encoded vector of a word to retrieve the vectors corresponding to its context words (words contained in the same sentence of the input word). This is know as **Skip-Gram** model and is the one adopted in this work as it performs better with small vocabularies (like logs' vocabulary).

Word2Vec

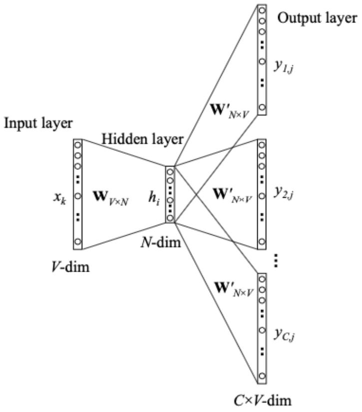


Figure: Skip-Gram model

Wor2Vec

Vectorization:

- applied on service level (so each logging service producing many types of messages)
- Results: vectors with length 100 for each word

Principal Component Analysis

PCA was used to:

- reduce dimensionality
- as an anomaly detection method itself trough decomposition and reconstruction

PCA Dimensionality Reduction

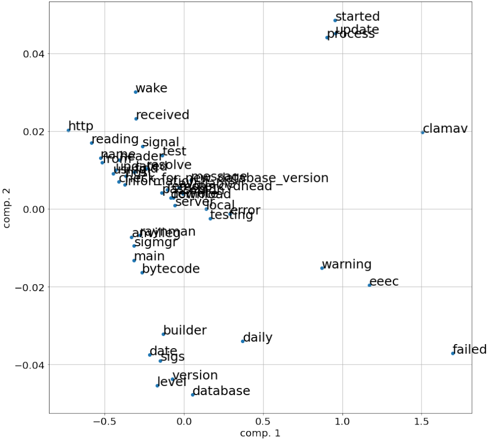


Figure: Example of Word Embedding with PCA dimensionality reduction with 2 components for the service **freshclam**

PCA Dimensionality Reduction

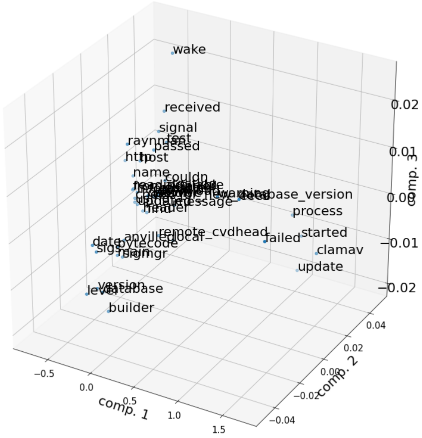


Figure: Example with 3 components

PCA And Reconstruction Error

- Process
  1. Reduce dimensionality trough PCA
  2. Do the inverse transformation using only the *n* principal components obtained to get an approximation of the initial data
  3. Compute **Reconstruction Error**
  4. Those vector-words for which the reconstruction error is larger are those identified as anomalies
- Rationale: It can be assumed that the reconstruction error is larger for uncommon, so less observed observations, for which the principal components of the sample are not able to explain most of their variability.

DBSCAN Clustering

Log Data

- Density based clustering method
- Epsilon obtained based on overall distances between word-vectors and parameter tuning
- Rationale: non-anomalous words are expected to be more and to concentrate closer between each other rather than potentially anomalous words.

Monitoring Data

- same rationale
- Epsilon obtained by means of elbow curve and and parameter tuning

DBSCAN Clustering

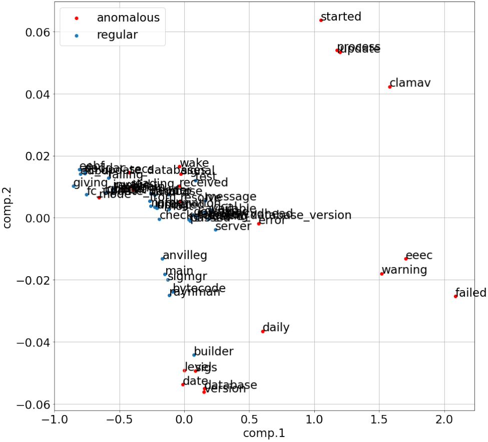


Figure: Example of DBSCAN for service Freshclam with PCA reduction to 2 components.

K-Means Clustering

- clustering method based on centroids
- number of clusters obtained with parameter tuning
- rationale: expected large distances between anomalous and non-anomalous word-vectors.

K-Means Clustering

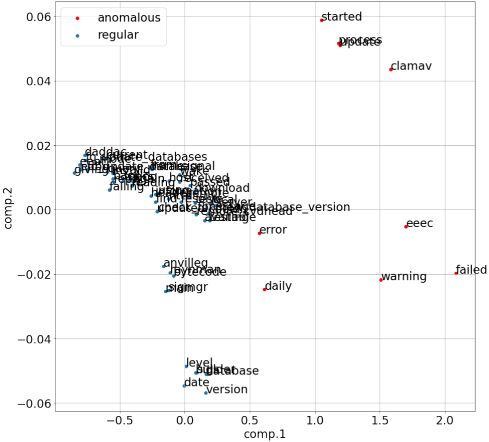


Figure: Example of K-Means with PCA reduction to 2 components and 2 clusters for the service freshclam.

Mean and Variance Outlier Detection

Log Data

- applied on machine level (so for every machine)
- Rationale: for each message:
  1. obtain the message-vector by averaging the vectors of all the words included in the message.
  2. calculate the distance between consecutive message-vectors.
  3. For those messages for which the distance is larger than a **threshold***, assign anomaly label
  4. Apply this process either on the full series or on **sliding windows**, so subsets of the series, making the analysis more precise on local anomalous behaviors.
  5. same rational on monitoring data vectors
- ***threshold** $= \mu + k\sigma$
  where $\mu = mean\ of\ sample$, $\sigma = s.d.\ of\ sample$ and k an integer (in literature it is usually equal to 1,2 or 3)

Mean and Variance Outlier Detection

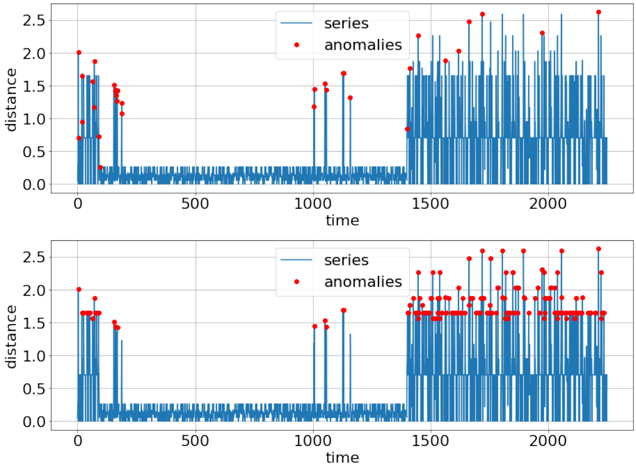


Figure: Anomaly detection with and without sliding windows, windows size=20 and tolerance=2 standard deviations on machine cloud-ctrl01.

Validating Results

1. **Log Data**: continous anomaly score $\in [0, 1]$ given to log messages. Applied after PCA, DBSCAN and K-Means algorithms results.
   - **PROS**: more robust and safe (e.g. misclassified words have less impact).
   - **CONS**: less precise and no more binary, intuitive classification.
2. **Monitoring Data**: hypothesis testing to check for differences between recognized as anomalous and non-anomalous samples.
   - this method does not provide information about the anomalous nature of data but at least it can be used to verify a significant separation between data significantly different from each other

Anomaly Score

daily database available for update (local version: 26052, remote version: 26053)

$$\Big\downarrow \text{ clean message}$$

'daily', 'database', 'available', 'for', 'update', 'local', 'version', 'remote', 'version'

$$1 + 0 + 0 + 0 + 1 + 0 + 0 + 0 + 0 = 2/9 = 0.222$$

Figure: Process to compute the anomaly score of a message

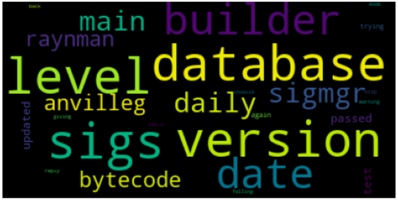
Hypothesis Testing

- Mann-Whitney U Test to check for significant differences between anomalous and non-anomalous classified observations, for every metric
- Those metrics with higher number of rejected hypothesis are those supposed to be the most critical ones (the ones causing more anomalies).

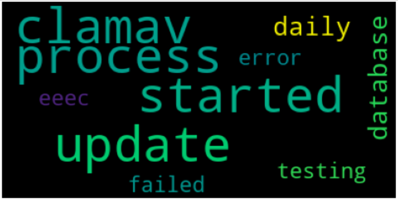
Algorithm	dim. reduction	n. comps	n. clusters	window size	Epsilon	tolerance
DBSCAN	TRUE, FALSE	2, 3, 10, 20	*variable*	*N.A.*	0.05%, 0.1%, 0.25%, 0.5%	*N.A.*
K-Means	TRUE, FALSE	2, 3, 10, 20	2, 3	*N.A.*	*N.A.*	*N.A.*
PCA Decomposition & Reconstruction	*implicit*	2, 3, 10, 20, 30	*N.A.*	*N.A.*	*N.A.*	0.75, 0.85, 0.95 (measured as the quantiles of the error vector in 3.4)
Time-Series Outlier Detection	*N.A.*	*N.A.*	*N.A.*	0, 10, 30, 60 (0 means no window partition)	*N.A.*	2, 3 (the k described in Equation 3.7, number of σ away from μ in the outlier detection algorithm)

Figure: Algorithms and Hyperparameters Overview

Keywords



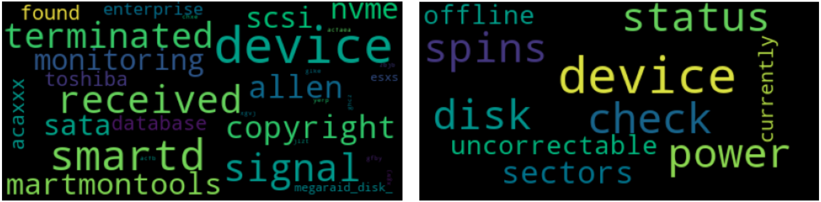
(a) Anomaly Score ≤ 0.3



(b) Anomaly Score ≥ 0.7

Figure: Wordcloud of keywords of messages from service **Freshclam** after **DBSCAN** Algorithm according to different anomaly scores

Keywords



(a) Anomaly Score ≤ 0.3 (b) Anomaly Score ≥ 0.7

Figure: Wordcloud of keywords of messages from service **Smartd** after **K-Means** Algorithm according to different anomaly scores

Keywords

(a) Anomaly Score ≤ 0.3 (b) Anomaly Score ≥ 0.7

Figure: Wordclouds of keywords of messages from service **storm-backend-stderr** after **PCA** Algorithm according to different anomaly scores

Main Challenges

- different type of data taken into consideration
- different algorithms and techniques adopted
- huge amount of data to deal with, automation was important but sometimes risky
- completely unsupervised task, so ad-hoc validation methods implemented

Future Developments

- hopefully in the future also some semi-supervised or even full supervised techniques might be employed
- feed together in a model textual and numerical data
- identify different types of anomalies and not only a binary classification (the anomaly score is a step forward in this regard)