On-line Bayesian Context Change Detection in Web Service Systems

Maciej Zięba, Jakub M. Tomczak

HotTopiCS 2013

maciej.zieba@spwr.wroc.pl

Prague, 20.04.2013
Agenda

1. Problem background
2. Change detection problem statement
3. Approaches for solving the problem
4. Bayesian Model
5. Algorithm for change detection
6. Simulation environment
7. Results and discussion
Problem background

SOA-based systems

- **SOA-based systems**: systems that implement Service Oriented Architecture, i.e., an architectural style whose goal is to achieve loose coupling among interacting software components called *services*.
Problem background

SOA-based systems

- **SOA-based systems**: systems that implement Service Oriented Architecture, i.e., an architectural style whose goal is to achieve loose coupling among interacting software components called *services*.

- **Services**: self-describing, stateless, modular applications that are distributed across the Web and which provide functionalities and are described by quality attributes (QoS).
Problem background

SOA-based systems

- **SOA-based systems**: systems that implement Service Oriented Architecture, i.e., an architectural style whose goal is to achieve loose coupling among interacting software components called services.

- **Execution system**:
  - network of virtual machines;
  - distributed computational resources;
  - input: streams of requests;
  - output: system performance, e.g., latency.
Problem background

Resource allocation problem

In order to maintain the performance of an execution system at a satisfactory (or given) level the following decisions are mainly be made:
Problem background
Resource allocation problem

- In order to maintain the performance of an execution system at a satisfactory (or given) level the following decisions are mainly be made:
  
  - migration of services;
In order to maintain the performance of an execution system at a satisfactory (or given) level the following decisions are mainly be made:

- migration of services;
- computational resources allocation.
Problem background

Resource re-allocation

- The execution system evolves in time because of:
Problem background

Resource re-allocation

- The execution system evolves in time because of:
  - non-stationary streams of requests;

![Diagram showing resource allocation over time](image).

n  n+1  n+2
Problem background

Resource re-allocation

- The execution system evolves in time because of:
  - non-stationary streams of requests;
  - failures or system’s modifications.

\[ n \rightarrow n+1 \rightarrow n+2 \]

Failure

![Diagram showing resource allocation changes over time with failure events.](image)
Problem background

Resource re-allocation

- The execution system evolves in time because of:
  - non-stationary streams of requests;
  - failures or system’s modifications.

- Hence, there is a need to propose an adaptive approach for resource allocation.
Problem background

Resource re-allocation

- The execution system evolves in time because of:
  - non-stationary streams of requests;
  - failures or system’s modifications.

Hence, there is a need to propose an **adaptive approach** for resource allocation.

- Resource re-allocation: if a *change* in the **input** (or **output**) is reported, then calculate new resource allocation.
Change detection problem

Overview

- **Change detection**: identifies changes in the probability distribution of a stochastic process.
Change detection problem

Overview

- **Change detection**: identifies changes in the probability distribution of a stochastic process.

- Two kinds of changes:
Change detection problem

Overview

- **Change detection**: identifies changes in the probability distribution of a stochastic process.

- Two kinds of changes:
  - gradual;

![Gradual change graph]
Change detection problem

Overview

- **Change detection**: identifies changes in the probability distribution of a stochastic process.

- Two kinds of changes:
  - gradual;
  - abrupt.
Change detection problem

Overview

- **Change detection**: identifies changes in the probability distribution of a stochastic process.

- Two kinds of changes:
  - gradual;
  - abrupt.

![Graph showing gradual and abrupt changes](image-url)
Change detection problem

Overview

- **Change detection**: identifies changes in the probability distribution of a stochastic process.

- **Two kinds of changes**: gradual; abrupt.

- **Statistical change detection**:
  - **frequentist approach**: distribution estimation and comparison using dissimilarity measures;
  - **Bayesian approach**: all quantities are random variables.

Diagram:
- Change detection
- Statistical inference
- Data
- Moments of change in streams of requests
Change detection problem

Overview

- **Change detection**: identifies changes in the probability distribution of a stochastic process.

- Two kinds of changes:
  - gradual;
  - abrupt.

- Statistical change detection:
  - **frequentist approach**: distribution estimation and comparison using dissimilarity measures;
  - **Bayesian approach**: all quantities are random variables.
Frequentist approach

Dissimilarity measure

Shifting window 1

Estimation

Shifting window 2

Estimation

Time
Change detection problem

Overview

- **Change detection**: tries to identify changes in the probability distribution of a stochastic process.

- Two kinds of changes:
  - gradual;
  - abrupt.

- Statistical change detection:
  - **frequentist approach**: distribution estimation and comparison using dissimilarity measures;
  - **Bayesian approach**: all quantities are random variables.
Bayesian approach

Likelihood functions for models

Assume that $D_n^L = \{x_{n-L+1}, \ldots, x_n\}$ are examples within shifting window of size $L$. 

1. If there is no context change in $D_n^L$, then we say that data are generated from a model $M_0$ and its likelihood function is as follows

$$p(D_n^L|M_0, \theta_0) = p(D_n^L|\theta_0) \tag{1}$$

where $\theta_0$ – parameters of $M_0$.

2. If there is one context change in $D_n^L$ at $t < n$, then we say that data are generated from a model $M_1$ and its likelihood function is as follows

$$p(D_n^L|M_1, \theta_1, t) = p(D_{n-t}^L|\theta_1) p(D_{n-t}^L|\theta_2) \tag{2}$$

where $\theta_1 = (\theta_1^1, \theta_2^1)^T$ – parameters of $M_1$, $\theta_1^1$ are parameters for partition before context change, and $\theta_2^1$ – parameters after context change.
Bayesian approach
Likelihood functions for models

Assume that $\mathcal{D}_n^L = \{x_{n-L+1}, \ldots, x_n\}$ are examples within shifting window of size $L$.

1. If there is no context change in $\mathcal{D}_n^L$, then we say that data are generated from a model $\mathcal{M}_0$ and its likelihood function is as follows

$$p(\mathcal{D}_n^L | \mathcal{M}_0, \theta_0) = p(\mathcal{D}_n^L | \theta_0)$$

where $\theta_0$ – parameters of $\mathcal{M}_0$. 

2. If there is one context change in $\mathcal{D}_n^L$ at $t<n$, then we say that data are generated from a model $\mathcal{M}_1$ and its likelihood function is as follows

$$p(\mathcal{D}_n^L | \mathcal{M}_1, \theta_1, t) = p(\mathcal{D}_{n-t}^L | \theta_1) p(\mathcal{D}_n^{t-1} | \theta_2)$$

where $\theta_1$ – parameters for partition before context change, and $\theta_2$ – parameters after context change.
Bayesian approach

Likelihood functions for models

Assume that $D_n^L = \{x_{n-L+1}, \ldots, x_n\}$ are examples within shifting window of size $L$.

1. If there is no context change in $D_n^L$, then we say that data are generated from a model $M_0$ and its likelihood function is as follows

$$ p(D_n^L | M_0, \theta_0) = p(D_n^L | \theta_0) \quad (1) $$

where $\theta_0$ – parameters of $M_0$.

2. If there is one context change in $D_n^L$ at $t < n$, then we say that data are generated from a model $M_1$ and its likelihood function is as follows

$$ p(D_n^L | M_1, \theta_1, t) = p(D_{t-n+t}^L | \theta_1^1) \cdot p(D_{n-t}^L | \theta_2^1) \quad (2) $$

where $\theta_1 = (\theta_1^1 \theta_1^2)^T$ – parameters of $M_1$, $\theta_1^1$ are parameters for partition before context change, and $\theta_1^2$ – parameters after context change.
Bayesian approach

Model evidence

In order to select one model which is *more probable* to generate observed data we need to calculate *model evidences*. The model evidence of $M_0$ can be calculated as follows
In order to select one model which is *more probable* to generate observed data we need to calculate *model evidences*. The model evidence of $\mathcal{M}_0$ can be calculated as follows

$$p(D^L_n | \mathcal{M}_0) = \int p(D^L_n | \mathcal{M}_0, \theta_0) \, p(\theta_0 | \mathcal{M}_0) \, d\theta_0,$$

where $p(\theta_0 | \mathcal{M}_0)$ – *a priori* probability distribution of parameters.
Bayesian approach

Model evidence

In order to select one model which is *more probable* to generate observed data we need to calculate *model evidences*. The model evidence of $M_0$ can be calculated as follows

$$ p(D_n^L | M_0) = \int p(D_n^L | M_0, \theta_0) \ p(\theta_0 | M_0) \ d\theta_0, \quad (3) $$

where $p(\theta_0 | M_0)$ – *a priori* probability distribution of parameters. Next, the model evidence of $M_1$ is the following (using the independence of $\theta_1^1, \theta_1^2, t$)

$$ p(D_n^L | M_1) = \iiint p(D_n^L | M_1, \theta_1, t) \ p(\theta_1^1 | M_1) \times $$

$$ \times p(\theta_1^2 | M_1) \ p(t | M_1) \ d\theta_1 \ dt, \quad (4) $$

where $p(\theta_1^1 | M_1), p(\theta_1^2 | M_1), p(t | M_1)$ – *a priori* probability distributions of parameters.
Bayesian approach

Model evidence approximation

In order to calculate model evidences of $\mathcal{M}_0$ and $\mathcal{M}_1$ we make the following assumptions:
In order to calculate model evidences of $M_0$ and $M_1$ we make the following assumptions:

- the *a priori* probability distributions of $\theta_0$ and $\theta_1$ are taken to be non-informative;
In order to calculate model evidences of $M_0$ and $M_1$ we make the following assumptions:

- the a priori probability distributions of $\theta_0$ and $\theta_1$ are taken to be non-informative;
- the context change occurs in the middle of the shifting window, i.e., $n - \left\lceil \frac{1}{2} L \right\rceil$, hence the a priori probability distribution of $t$ is a Dirac delta function in the point $n - \left\lceil \frac{1}{2} L \right\rceil$. 
Bayesian approach
Model evidence approximation

In order to calculate model evidences of $\mathcal{M}_0$ and $\mathcal{M}_1$ we make the following assumptions:

- the \textit{a priori} probability distributions of $\theta_0$ and $\theta_1$ are taken to be non-informative;
- the context change occurs in the middle of the shifting window, i.e., $n - \left\lceil \frac{1}{2} L \right\rceil$, hence the \textit{a priori} probability distribution of $t$ is a Dirac delta function in the point $n - \left\lceil \frac{1}{2} L \right\rceil$.

For such assumptions we can approximate the model evidence by the Bayesian Information Criterion (BIC)

$$
\ln p(D_n^L | \mathcal{M}) \approx \ln p(D_n^L | \hat{\theta}) - \frac{K}{2} \ln L, \tag{5}
$$

where $\hat{\theta}$ is the maximum likelihood estimator of $\theta$. 

To compare both models, we calculate the Bayes factor (assuming equal probabilities over models):

\[
B_{10} = \frac{p(D^n_L|M_1)}{p(D^n_L|M_0)}.
\] (6)

<table>
<thead>
<tr>
<th>(B_{10})</th>
<th>(\ln(B_{10}))</th>
<th>Evidence in favor of (M_1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 – 3</td>
<td>0 – 1.1</td>
<td>Weak</td>
</tr>
<tr>
<td>3 – 10</td>
<td>1.1 – 2.3</td>
<td>Substantial</td>
</tr>
<tr>
<td>10 – 100</td>
<td>2.3 – 4.6</td>
<td>Strong</td>
</tr>
<tr>
<td>&gt; 100</td>
<td>&gt; 4.6</td>
<td>Decisive</td>
</tr>
</tbody>
</table>
Algorithm description

**Algorithm 1**: Change detection using approximated Bayes factor

**Input**: $\mathcal{D}$, $L$, $\mathcal{M}_0$, $\mathcal{M}_1$

**Output**: Moments of context change $\tau_1, \ldots, \tau_M$

1. $n \leftarrow 1$, $m \leftarrow 0$, $\tau_0 \leftarrow 0$;
2. while $n < \text{card}\{\mathcal{D}\}$ do
3.   Calculate $\ln p(\mathcal{D}_{n}^L | \mathcal{M}_0)$ and $\ln p(\mathcal{D}_{n}^L | \mathcal{M}_1)$;
4.   Calculate $\ln B_{10}$;
5.   if $\ln B_{10} > \sigma$ then
6.     if $((n - \lceil L/2 \rceil) - \tau_m) \geq \lfloor L/2 \rfloor$ then
7.       $m \leftarrow m + 1$;
8.     $\tau_m \leftarrow n - \lceil L/2 \rceil$;
9.   end
10. end
11. $n \leftarrow n + 1$;
12. end
Simulator

Structure

Request generator -> Scheduler

Scheduler -> Computational unit

Computational unit -> Sink

Computational unit

Virtual machine

Web service

Web service

Virtual machine

Web service

Web service
Streams of requests are generated with **Poisson process**.
Simulator

Details

- Streams of requests are generated with Poisson process.
- Computational nodes are represented by web servers with processors as computational resources.
Simulator

Details

- Streams of requests are generated with **Poisson process**.

- Computational nodes are represented by web servers with **processors** as computational resources.

- Two virtual machines are situated on each of servers.
Simulator

Details

- Streams of requests are generated with Poisson process.
- Computational nodes are represented by web servers with processors as computational resources.
- Two virtual machines are situated on each of servers.
- Each of two web servers in the model uses 8 processors, which are assigned to virtual machines in following way:
Simulator

Details

- Streams of requests are generated with Poisson process.

- Computational nodes are represented by web servers with processors as computational resources.

- Two virtual machines are situated on each of servers.

- Each of two web servers in the model uses 8 processors, which are assigned to virtual machines in following way:

  - 6 and 2 processors are respectively used by first and second virtual machine (first server).
Simulator
Details

- Streams of requests are generated with **Poisson process**.

- Computational nodes are represented by web servers with **processors** as computational resources.

- Two virtual machines are situated on each of servers.

- Each of two web servers in the model uses 8 **processors**, which are assigned to virtual machines in following way:

  - 6 and 2 processors are respectively used by **first** and **second virtual machine** (first server).

  - 4 and 4 processors are respectively used by **first** and **second virtual machine** (second server).
Simulator

Details

- Streams of requests are generated with Poisson process.
- Computational nodes are represented by web servers with processors as computational resources.
- Two virtual machines are situated on each of servers.
- Each of two web servers in the model uses 8 processors, which are assigned to virtual machines in following way:
  - 6 and 2 processors are respectively used by first and second virtual machine (first server).
  - 4 and 4 processors are respectively used by first and second virtual machine (second server).
- Processing delays for web servers are equal 0.0004 seconds and for virtual machines are equal 0.0008 seconds

According to the technical report: Lite Technologies, Web server performance comparison: Litespeed 2.0 vs..
Simulator
Modelling Web services

- Performance of real data processing services implemented in PlaTel was modelled: Naive Bayes, Logistic Regression, J48 and Multilayer Perceptron.
Performance of real data processing services implemented in PlaTel was modelled: *Naive Bayes, Logistic Regression, J48* and *Multilayer Perceptron*.

Processing time for each of selected services was modelled with *triangular distribution*. 
Performance of real data processing services implemented in PlaTel was modelled: *Naive Bayes, Logistic Regression, J48* and *Multilayer Perceptron*.

Processing time for each of selected services was modelled with *triangular distribution*.

The values parameters for distributions (minimum, maximum and average value) for each of services were estimated using *soapUI tool*. 
Simulator
Modelling Web services

- Performance of real data processing services implemented in PlaTel was modelled: *Naive Bayes*, *Logistic Regression*, *J48* and *Multilayer Perceptron*.

- Processing time for each of selected services was modelled with **triangular distribution**.

- The values parameters for distributions (minimum, maximum and average value) for each of services were estimated using *soapUI tool*.

- Following resource allocation of web services were proposed:
Performance of real data processing services implemented in PlaTel was modelled: Naive Bayes, Logistic Regression, J48 and Multilayer Perceptron.

Processing time for each of selected services was modelled with triangular distribution.

The values parameters for distributions (minimum, maximum and average value) for each of services were estimated using soapUI tool.

Following resource allocation of web services were proposed:

- Multilayer Perceptron - total number of 10 processors.
Performance of real data processing services implemented in PlaTel was modelled: *Naive Bayes, Logistic Regression, J48* and *Multilayer Perceptron*.

Processing time for each of selected services was modelled with **triangular distribution**.

The values parameters for distributions (minimum, maximum and average value) for each of services were estimated using *soapUI* tool.

Following resource allocation of web services were proposed:

- **Multilayer Perceptron** - total number of 10 processors.
- **Logistic Regression** total number of 10 processors.
Simulator
Modelling Web services

- Performance of real data processing services implemented in PlaTel was modelled: *Naive Bayes*, *Logistic Regression*, *J48* and *Multilayer Perceptron*.

- Processing time for each of selected services was modelled with **triangular distribution**.

- The values parameters for distributions (minimum, maximum and average value) for each of services were estimated using **soapUI tool**.

- Following resource allocation of web services were proposed:
  - *Multilayer Perceptron* - total number of **10 processors**.
  - *Logistic Regression* total number of **10 processors**.
  - *J48* total number of **6 processors**.
Simulator
Modelling Web services

- Performance of real data processing services implemented in PlaTel was modelled: Naive Bayes, Logistic Regression, J48 and Multilayer Perceptron.

- Processing time for each of selected services was modelled with triangular distribution.

- The values parameters for distributions (minimum, maximum and average value) for each of services were estimated using soapUI tool.

- Following resource allocation of web services were proposed:
  - Multilayer Perceptron - total number of 10 processors.
  - Logistic Regression total number of 10 processors.
  - J48 total number of 6 processors.
  - Naive Bayes total number of 4 processors.
The aim of the experiment is to compare the performance of frequentist approaches with Bayesian approach for change detection problem.
The aim of the experiment is to compare the performance of frequentist approaches with Bayesian approach for change detection problem.

Following dissimilarity measures were considered:

- Bhattacharyya
- Kullback-Leibler
- Lin-Wong
- Modified Lin-Wong

Average latency in request responses was considered as a quality rate for entire system.

The simulation model was implemented in a discrete event simulation environment named Arena.

Algorithms for change detection were implemented in Matlab.
Experiment

Description

- The aim of the experiment is to compare the performance of frequentist approaches with Bayesian approach for change detection problem.

- Following dissimilarity measures were considered:
  - Bhattacharyya
Experiment

Description

- The aim of the experiment is to compare the performance of frequentist approaches with Bayesian approach for change detection problem.

- Following dissimilarity measures were considered:
  - Bhattacharyya
  - Kullback-Leibler
The aim of the experiment is to compare the performance of frequentist approaches with Bayesian approach for change detection problem.

Following dissimilarity measures were considered:

- Bhattacharyya
- Kullback-Leibler
- Lin-Wong
The aim of the experiment is to compare the performance of frequentist approaches with Bayesian approach for change detection problem.

Following dissimilarity measures were considered:

- Bhattacharyya
- Kullback-Leibler
- Lin-Wong
- modified Lin-Wong

Average latency in request responses was considered as a quality rate for entire system.

The simulation model was implemented in discrete events simulation environment Arena. Algorithms for change detection were implemented in Matlab.
The aim of the experiment is to compare the performance of frequentist approaches with Bayesian approach for change detection problem.

Following dissimilarity measures were considered:

- Bhattacharyya
- Kullback-Leibler
- Lin-Wong
- modified Lin-Wong

Average latency in request responses was considered as a quality rate for entire system.
The aim of the experiment is to compare the performance of frequentist approaches with Bayesian approach for change detection problem.

Following dissimilarity measures were considered:

- Bhattacharyya
- Kullback-Leibler
- Lin-Wong
- modified Lin-Wong

Average latency in request responses was considered as a quality rate for entire system.

The simulation model was implemented in discrete events simulation environment Arena.
The aim of the experiment is to compare the performance of frequentist approaches with Bayesian approach for change detection problem.

Following dissimilarity measures were considered:

- Bhattacharyya
- Kullback-Leibler
- Lin-Wong
- modified Lin-Wong

Average latency in request responses was considered as a quality rate for entire system.

The simulation model was implemented in discrete events simulation environment Arena.

Algorithms for change detection were implemented in Matlab.
Experiment
Considered scenarios (1)

1. *Slight context change*. The context is changed periodically (5 times per simulation) and change is gained by increasing the intensity parameters of Poisson process three times.

2. *Significant context change*. The context is changed periodically (5 times per simulation) and change is gained by increasing the intensity parameters of Poisson process six times.
3. **Processors failure (anomaly).** Anomaly is gained by failure of 4 processors on first virtual machine.
## Experiment

Results for slight context change simulation

<table>
<thead>
<tr>
<th>Measure</th>
<th>Correctly detected (max. 5)</th>
<th>Incorrectly detected</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bhattacharyya</td>
<td></td>
<td></td>
</tr>
<tr>
<td>($L = 25, \sigma = 0.2$)</td>
<td>3.2</td>
<td>0.2</td>
</tr>
<tr>
<td>Kullback-Leibler</td>
<td></td>
<td></td>
</tr>
<tr>
<td>($L = 25, \sigma = 1$)</td>
<td>3.8</td>
<td>0.8</td>
</tr>
<tr>
<td>Lin-Wong</td>
<td></td>
<td></td>
</tr>
<tr>
<td>($L = 25, \sigma = 0.15$)</td>
<td>2.8</td>
<td>0.7</td>
</tr>
<tr>
<td>mod. Lin-Wong</td>
<td></td>
<td></td>
</tr>
<tr>
<td>($L = 25, \sigma = 0.02$)</td>
<td>2.9</td>
<td>0.9</td>
</tr>
<tr>
<td>Bayesian approach</td>
<td></td>
<td></td>
</tr>
<tr>
<td>($L = 25$)</td>
<td>3</td>
<td>0.2</td>
</tr>
</tbody>
</table>
## Experiment
Results for significant context change simulation

<table>
<thead>
<tr>
<th>Measure</th>
<th>Correctly detected (max. 5)</th>
<th>Incorrectly detected</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bhattacharyya</td>
<td></td>
<td></td>
</tr>
<tr>
<td>((L = 25, \sigma = 0.2))</td>
<td>4.6</td>
<td>0.1</td>
</tr>
<tr>
<td>Kullback-Leibler</td>
<td></td>
<td></td>
</tr>
<tr>
<td>((L = 25, \sigma = 1))</td>
<td>4.8</td>
<td>0.2</td>
</tr>
<tr>
<td>Lin-Wong</td>
<td></td>
<td></td>
</tr>
<tr>
<td>((L = 25, \sigma = 0.15))</td>
<td>4.6</td>
<td>0.3</td>
</tr>
<tr>
<td>mod. Lin-Wong</td>
<td></td>
<td></td>
</tr>
<tr>
<td>((L = 25, \sigma = 0.02))</td>
<td>4.6</td>
<td>0.2</td>
</tr>
<tr>
<td>Bayesian approach</td>
<td></td>
<td></td>
</tr>
<tr>
<td>((L = 25))</td>
<td>5</td>
<td>0</td>
</tr>
</tbody>
</table>
## Experiment

Results for processors failure simulation

<table>
<thead>
<tr>
<th>Measure</th>
<th>Correctly detected (max. 2)</th>
<th>Incorrectly detected</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bhattacharyya</td>
<td>1</td>
<td>0.3</td>
</tr>
<tr>
<td>(L = 25, \sigma = 0.2)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Kullback-Leibler</td>
<td>0.7</td>
<td>0.3</td>
</tr>
<tr>
<td>(L = 25, \sigma = 1)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lin-Wong</td>
<td>1.1</td>
<td>0.1</td>
</tr>
<tr>
<td>(L = 25, \sigma = 0.15)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>mod. Lin-Wong</td>
<td>1</td>
<td>0.1</td>
</tr>
<tr>
<td>(L = 25, \sigma = 0.02)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bayesian approach</td>
<td>1.1</td>
<td>0.1</td>
</tr>
<tr>
<td>(L = 25)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Discussion

- Most of changes were successfully detected using frequentist and Bayesian approaches.
Discussion

- Most of changes were successfully detected using frequentist and Bayesian approaches.

- The lowest number of detected changes were gained for *Slight context change* and *Processors failure* simulation scenarios.

- Bayesian approach, in comparison to the frequentist approach, does not demand defining additional parameters beside shifting window’s size.
Discussion

- Most of changes were successfully detected using frequentist and Bayesian approaches.

- The lowest number of detected changes were gained for *Slight context change* and *Processors failure* simulation scenarios.

- Bayesian approach performed slightly better for *Significant context change* and *Processors failure (anomaly)* scenarios.
Discussion

- Most of changes were successfully detected using frequentist and Bayesian approaches.

- The lowest number of detected changes were gained for *Slight context change* and *Processors failure* simulation scenarios.

- Bayesian approach performed slightly better for *Significant context change* and *Processors failure (anomaly)* scenarios.

- The number of incorrectly detected changes using Bayesian model was the lowest for all considered scenarios.
Discussion

- Most of changes were successfully detected using frequentist and Bayesian approaches.

- The lowest number of detected changes were gained for *Slight context change* and *Processors failure* simulation scenarios.

- Bayesian approach performed slightly better for *Significant context change* and *Processors failure (anomaly)* scenarios.

- The number of incorrectly detected changes using Bayesian model was the lowest for all considered scenarios.

- The best results for slight context changes were gained using Bhattacharyya measure.
Discussion

- Most of changes were successfully detected using frequentist and Bayesian approaches.

- The lowest number of detected changes were gained for *Slight context change* and *Processors failure* simulation scenarios.

- Bayesian approach performed slightly better for *Significant context change* and *Processors failure (anomaly)* scenarios.

- The number of incorrectly detected changes using Bayesian model was the lowest for all considered scenarios.

- The best results for slight context changes were gained using Bhattacharyya measure.

- Bayesian approach, in comparison to the frequentist approach, does not demand defining additional parameters beside shifting window’s size.