PONTIFÍCIA UNIVERSIDADE CATÓLICA DO PARANÁ

PRÓ-REITORIA DE PESQUISA, PÓS-GRADUAÇÃO E Inovação  
PROGRAMA INSTITUCIONAL DE BOLSAS DE INICIAÇÃO CIENTÍFICA

PIBIC



**RELATÓRIO FINAL**

##### DESENVOLVIMENTO DE APLICAÇÃO PARA PRÉ-PROCESSAMENTO DE LOG DE EVENTOS NA ÁREA DA SAÚDE

**Edson Emilio Scalabrin**

**CURITIBA**

**30 de junho de 2021**

**ANDERSON FUZINO**

**EDSON EMILIO SCALABRIN**

**LUIZ FERNANDO PUTTOW SOUTHIER (colaborador)**

**CIÊNCIA DA COMPUTAÇÃO**

**PUCPR**

**DESENVOLVIMENTO DE APLICAÇÃO PARA PRÉ-PROCESSAMENTO DE LOG DE EVENTOS NA ÁREA DA SAÚDE**

Relatório Final apresentado ao Programa Institucional de Programa Institucional de Bolsas de Iniciação Científica e Tecnológica, Pró-Reitoria de Pesquisa, Pós-Graduação e Inovação da Pontifícia Universidade Católica do Paraná, sob orientação do:

Prof. Edson Emilio Scalabrin e

Msc. Luiz Southier – colaborador (doutorando)

**SUMÁRIO**

[1. INTRODUÇÃO 1](#_Toc75960233)

[2. OBJETIVOS 2](#_Toc75960234)

[2.1 Objetivos específicos 2](#_Toc75960235)

[3. REVISÃO DE LITERATURA 2](#_Toc75960236)

[4. MATERIAIS E MÉTODO 3](#_Toc75960237)

[5. RESULTADOS 3](#_Toc75960238)

[5.1. Granularidade 4](#_Toc75960239)

[5.2. Valores ausentes 4](#_Toc75960240)

[5.3. Ordem lógica 6](#_Toc75960241)

[6. DISCUSSÃO 7](#_Toc75960242)

[7. CONSIDERAÇÕES FINAIS 7](#_Toc75960243)

[REFERÊNCIAS 7](#_Toc75960244)

**RESUMO**

**Introdução:** A mineração de processos tem sido aplicada com sucesso em diversas áreas, incluindo serviços bancários, produção, educação e saúde. Ela relaciona o comportamento real de pessoas, máquinas e organizações ao comportamento modelado. Tal relacionamento muitas vezes leva a descobertas surpreendentes colocando em evidência que a realidade é muito diferente das percepções, opiniões e crenças das partes interessadas. Para isso, a mineração de processos utiliza um log de eventos que registra informações sobre os eventos do processo, tais como, timestamp, atividade, caso, recursos utilizados, entre outras. Porém, o log criado deve passar por verificações de qualidade a fim de se determinar se devem ser aplicadas técnicas de filtragem, enriquecimento ou correção de erros. **Objetivos:** Almeja-se construir uma aplicação em software para auxiliar e facilitar o processo de avaliação de log de eventos. Essa aplicação deve dar suporte a profissionais, não necessariamente programadores, para que realizem o pré-processamento. **Materiais e Método:** A literatura referente à avaliação de log de eventos foi explorada e diversas técnicas de avaliação foram identificadas e estudadas. Três técnicas de avaliação foram selecionadas para implementação. A implementação se deu em ambiente python de maneira modular, isto é, cada técnica foi implementada e testada separadamente. Para testes e ajustes foram utilizados logs de eventos artificiais. **Resultados:** Três módulos foram desenvolvidos, uma para cada técnica de avaliação selecionada. A técnica de granularidade identifica para cada atividade qual é a granularidade temporal mínima. Possíveis discrepâncias entre granularidade de atividades podem ser identificadas e quantificadas com essa técnica. A técnica de detecção de valores ausentes quantifica para o log, para cada atividade e para cada caso, quantos valores ausentes são identificados. A técnica de verificação de violação de ordem lógica, parte da criação de uma regra de ordem entre duas atividades. Todos os casos que não respeitam essa ordem, são identificados. Os detalhes das entradas e saídas de cada módulo, bem como o código-fonte, são apresentados. **Considerações Finais:** Espera-se que o trabalho desenvolvido colabore para a avaliação da qualidade de log de eventos. Como perspectiva de trabalhos futuros, tem-se a implementação de outras técnicas de pré-processamento de logs de eventos.

**Palavras-chave**: Mineração de Processos. Avaliação de log de eventos. Pré-processamento.

# **INTRODUÇÃO**

Atualmente, observa-se que há uma pressão crescente sobre as organizações de saúde. Elas precisam melhorar a produtividade e reduzir a entrada no sistema e os tempos de espera; o que deve diminuir tempo e custos. Uma abordagem é focar nos processos de longo termo e não triviais que são realizados, por exemplo, no interior de hospitais (e.g., preparação e execução de uma cirurgia, tratamento de pacientes com câncer). A impressão de qualidade e satisfação de um paciente está diretamente ligada ao seu tempo de espera e permanência no percurso de tratamento. Portanto, para dar sugestões objetivas para melhorar e redesenhar os processos, é necessário explorar os dados do evento prontamente disponíveis. Tal análise é possível usando mineração de processo (AALST, 2016).

A mineração de processos foi aplicada com sucesso em diversas áreas, incluindo serviços bancários, produção, educação, diagnósticos inteligentes etc. (GARCIA et al., 2019). A mineração de processos relaciona o comportamento real de pessoas, máquinas e organizações ao comportamento modelado (AALST; WEIJTERS, 2004). Tal relacionamento muitas vezes leva a descobertas surpreendentes colocando em evidência que a realidade é muito diferente das percepções, opiniões e crenças das partes interessadas.

Para isso, a mineração de processos utiliza um log de eventos que é um arquivo que registra informações sobre os eventos do processo, tais como, *timestamp*, atividade, caso, recursos utilizados, entre outras. (AALST, 2011; BATISTA; SOLANAS, 2018). Porém, muitas vezes a obtenção do log das bases de dados é uma tarefa de alta complexidade, visto que tais bases podem se apresentar das mais diversas maneiras. Ainda, o log criado com as informações resgatas das bases deve passar por verificações de qualidade a fim de se determinar se devem ser aplicadas técnicas de filtragem, enriquecimento ou correção de erros (VANBRABANT, 2019; SURIADI, 2017). Assim, o processo de verificação de qualidade do log de eventos é de extrema necessidade para a adequada execução do pré-processamento do log.

# **OBJETIVOS**

Almeja-se construir uma aplicação em software para auxiliar e facilitar o processo de avaliação de log de eventos. Essa aplicação deve dar suporte a profissionais, não necessariamente programadores, para que realizem o pré-processamento.

## 2.1 Objetivos específicos

1. Realizar um estudo da literatura sobre métodos de avaliação de dados de sistemas de informações com vista a construção de logs de eventos;

2. Selecionar um subconjunto de métodos de avaliação de logs de eventos;

3. Implementar o subconjunto de métodos selecionados; e

4. Testar os métodos implementados.

# **REVISÃO DE LITERATURA**

Dixit (2018) apresenta técnicas de avaliação e detecção de problemas na ordem dos eventos em um determinado log. A abordagem proposta neste artigo visa detectar e reparar imperfeições de ordenação de eventos que surgem devido a problemas relacionados ao timestamp. Como primeira etapa, os pesquisadores detectam exatamente onde podem existir problemas de carimbo de timestamp em um log, reconhecendo as características comumente associadas à imperfeição de ordenação de eventos.

Três técnicas de avaliação de timestamp são apresentadas. A primeira está relacionada à existência de granularidade grosseira em timestamp ou granularidade de timestamp mista (por exemplo, log de eventos inclui eventos de vários sistemas onde cada sistema registra timestamps de forma diferente). A segunda descreve a detecção de eventos que exibem uma ordem de acontecimentos incomum ou eventos repetidos. A terceira técnica detecta anomalias estatísticas mais genéricas, como aprender a posição temporal de uma determinada atividade no contexto de outras atividades, ou a distribuição de valores de timestamp de todos os eventos em um log.

Vanbrabant (2019) propõe diversas técnicas de avaliação de log de eventos. As técnicas estão relacionadas à detecção de dados faltantes como atributos, entidades e valores. Além disso, são explorados problemas de informação errada, como violação de ordem lógica, violação de dependência mútua, inexatidão de timestamp, erros de digitação ou valores fora de intervalos pré-definidos. Ainda, são detectados problemas de formatação incorreta nos dados, valores não explícitos, abreviações, entre outros.

Em ambos os artigos são apresentadas técnicas para análise da qualidade do log de eventos.

# **MATERIAIS E MÉTODO**

Para estudos, avaliação e seleção de técnicas de log de eventos utilizou-se os trabalhos de Dixit (2018) e Vanbrabant (2019). Dentre as técnicas apresentadas nestes trabalhos, algumas foram selecionadas para implementação:

● Detecção de diferentes granularidades de timestamp;

● Detecção de valores ausentes; e

● Detecção de violação de ordem lógica.

Neste projeto utilizou-se para implementação o ambiente do COLAB (python). Cada técnica foi implementada separadamente em um script do COLAB. Utilizou-se um log de eventos artificial para avaliação das técnicas implementadas.

# **RESULTADOS**

Como principal resultado, estão os módulos implementados. A seguir serão apresentados os três módulos desenvolvidos. Descreve-se para cada módulo as entradas e saídas.

## Granularidade

Este módulo implementa a técnica de detecção de diferentes granularidades de timestamp. Para isso as entradas do módulo são:

* **Log de eventos** em formato *csv.* O log de eventos é composto por um registro de eventos incluindo a informação do caso a qual o evento pertence, qual atividade está sendo executada, o *timestamp* desta atividade, e outras informações extras.
* **Formato do timestamp** presente no log de eventos. Por exemplo, %d/%m/%Y %H:%M representa que o *timestamp* no log está definido em formato de dia/mês/ano hora:minutos.
* Nome da **coluna de atividade** no log de eventos.
* Nome da **coluna de** **timestamp** no log de eventos.

Com as informações de entrada, verifica-se para cada atividade a quantidade de eventos existentes. Para cada evento verifica-se a menor granularidade presente no timestamp. A granularidade pode ser em segundos, minutos, horas, dias, meses ou anos. Em seguida, são agregadas as quantidade, para cada atividade e para cada granularidade diferente. Os resultados são apresentados tabularmente, como no exemplo a seguir:

|  |
| --- |
| ------------- -------- ------- ----- ---- ----- ----  0 Segundos Minutos Horas Dias Meses Anos  Alta paciente 0 0 0 4 0 0  Triagem 0 1 4 0 0 0  Consulta 0 3 2 0 0 0  Exame 0 3 0 0 0 0  ------------- -------- ------- ----- ---- ----- ---- |

## Valores ausentes

Este módulo implementa a técnica de detecção de valores ausentes. Para isso as entradas do módulo são:

* **Log de eventos** em formato *csv.* O log de eventos é composto por um registro de eventos incluindo a informação do caso a qual o evento pertence, qual atividade está sendo executada, o *timestamp* desta atividade, e outras informações extras.
* Nome da **coluna de atividade** no log de eventos.
* Nome da **coluna de identificação de caso** no log de eventos.
* Representação do **valor ausente** indicando como esse tipo de valor é representado no log. Por exemplo, valores ausentes podem ser representados com um caractere de “-”.

Com as informações de entrada, verifica-se para todo o log, para cada atividade e para cada caso a quantidade de valores ausentes. Os resultados são apresentados textualmente. Para o log, por exemplo:

|  |
| --- |
| Análise do log  Nulos Total Porcentagem  14 68 20.588235294117645 |

Para atividades, por exemplo:

|  |
| --- |
| Análise por atividade  Alta paciente  Nulos Total Porcentagem  1 4 25.0  -  Nulos Total Porcentagem  2 2 100.0  Triagem  Nulos Total Porcentagem  3 4 75.0 |

Para casos, por exemplo:

|  |
| --- |
| Análise por caso  Caso 1  Nulos Total Porcentagem  2 3 66.66666666666666  Caso 2  Nulos Total Porcentagem  4 4 100.0  Caso 3  Nulos Total Porcentagem  1 4 25.0 14 68 20.588235294117645 |

## Ordem lógica

Este módulo implementa a técnica de detecção de violação de ordem lógica. Nessa técnica, verifica-se se uma certa atividade B, que só pode ocorrer após uma atividade A, ocorreu no log sem a presença anterior de A. Isto é, A é requisito para a ocorrência de B, nessa ordem: primeiro A, e depois B.

Para isso as entradas do módulo são:

* **Log de eventos** em formato *csv.* O log de eventos é composto por um registro de eventos incluindo a informação do caso a qual o evento pertence, qual atividade está sendo executada, o *timestamp* desta atividade, e outras informações extras.
* Nome da **coluna de atividade** no log de eventos.
* Nome da **coluna de identificação de caso** no log de eventos.
* Nome da **coluna de** **timestamp** no log de eventos.
* Nome da **atividade A**
* Nome da **atividade B** que só pode ocorrer em um momento depois de A.

Com as informações de entrada, verifica-se para cada caso no log se houve violações da ordem estabelecida: B ocorrer depois de A. Os resultados são apresentados textualmente:

|  |
| --- |
| QuantidadeViolacoes TotalCasos Porcentagem  1 4 25.0 |

# **DISCUSSÃO**

Utilizando os módulos implementados é possível identificar problemas de qualidade no log de evento dado como entrada para estes módulos. Problemas de diferença de granularidade podem ser detectados, identificando atividades que são geradas em uma maior granularidade, do que se comparado ao restante do log. No segundo módulo, é possível observar o percentual de valores ausentes/nulos no log de eventos, identificando possíveis erros de importação, por exemplo. Ainda, uma regra de ordem entre duas atividades pode ser definida e verificada em comparação ao log.

# **CONSIDERAÇÕES FINAIS**

Este trabalho objetivou o desenvolvimento de uma aplicação para pré-processamento de log de eventos. Desenvolveu-se três módulos com técnicas de avaliação da qualidade dos logs de eventos. Os módulos foram testados com logs artificiais. Para pesquisas futuras, pretende-se implementar outras técnicas de pré-processamento, além das três aqui apresentadas.

# **REFERÊNCIAS**

AALST, W. V. D. Process mining: discovery, conformance and enhancement of business processes. Heidelberg: Springer, 2011.

AALST, W. V. D. et al. Process mining manifesto. In: International Conference on Business Process Management. Springer, Berlin, Heidelberg, 2011. p. 169-194.

AALST, W. V. D. Data science in action. In: Process mining. [S.l.]: Springer, 2016. p. 3–23.

AALST, W. M. Van der; WEIJTERS, A. J. Process mining: a research agenda. [S.l.]: Elsevier, 2004.

BATISTA, E.; SOLANAS, A. Process Mining in Healthcare: A Systematic Review. In: 2018 9th International Conference on Information, Intelligence, Systems and Applications (IISA). [S.l.: s.n.], 2018. p. 1–6

DIXIT, Prabhakar M. et al. Detection and interactive repair of event ordering imperfection in process logs. In: International Conference on Advanced Information Systems Engineering. Springer, Cham, 2018. p. 274-290.

GARCIA, C. d. S. et al. Process mining techniques and applications – A systematic mapping study. Expert Systems with Applications, v. 133, p. 260–295, nov. 2019. ISSN 0957-4174.

SURIADI, S. et al. Event log imperfection patterns for process mining: Towards a systematic approach to cleaning event logs. Information Systems, v. 64, p. 132–150, mar. 2017. ISSN 0306-4379

VANBRABANT, L. et al. Quality of input data in emergency department simulations: Framework and assessment techniques. Simulation Modelling Practice and Theory, v. 91, p. 83–101, fev. 2019. ISSN 1569190X.

# **ANEXO I**

Os códigos fontes para os três módulos são aqui apresentados.

|  |
| --- |
| # -\*- coding: utf-8 -\*-  """Tabela Granularidade.ipynb  Automatically generated by Colaboratory.  Original file is located at  https://colab.research.google.com/drive/15yjZOAyiB77KFw2t04111JWFRDrTvqhh  """  ## informações que o usuário vai fornecer  import pandas as pd  import numpy as np  from tabulate import tabulate  import datetime as dt  # import qadapo as vio  #importação do log de eventos - planilha d excel  log = pd.read\_excel('ConsultarGranularidade.xlsx', index\_col=0)  # formato do timestamp  formatoTimeStamp = '%d/%m/%Y %H:%M'  # nome da coluna de atividades  ColunaAtividade = 'Atividade/evento'  # nome da coluna timestamp  ColunaTimestamp = 'Timestamp'  # sao os nomes da granularidade  rGranularidade = ['Segundos','Minutos','Horas','Dias','Meses','Anos']  listadeatividades = log[ColunaAtividade].unique()  TabelaFinal = [[0 for i in range(len(rGranularidade) + 1)] for j in range(len(listadeatividades) + 1)]  for i in range(len(rGranularidade)):  TabelaFinal[0][i + 1] = rGranularidade[i]  for i in range(len(listadeatividades)):  TabelaFinal[i + 1][0] = listadeatividades[i]  print(tabulate(TabelaFinal))  j = 0  for a in listadeatividades:  j = j + 1  for i, linha in log.iterrows():  # dentro desse laço cada linha da tabela LOG está na variável LINHA  # essa função captura o timestamp na variável TIMESTAMP  timestamp = dt.datetime.strptime(linha[ColunaTimestamp], formatoTimeStamp)  if a == linha[ColunaAtividade]:  # verifica se os segundos do timestamp sao diferentes de zero  if(timestamp.second != 0):  TabelaFinal[j][1] += 1    # verifica se os minutos do timestamp sao diferentes de zero  elif(timestamp.minute != 0):  TabelaFinal[j][2] += 1  # verifica se os horas do timestamp sao diferentes de zero  elif(timestamp.hour != 0):  TabelaFinal[j][3] += 1  # verifica se os dias do timestamp sao diferentes de zero  elif(timestamp.day != 0):  TabelaFinal[j][4] += 1  # verifica se os meses do timestamp sao diferentes de zero  elif(timestamp.month != 0):  TabelaFinal[j][5] += 1  # verifica se os anos do timestamp sao diferentes de zero  elif(timestamp.year != 0):  TabelaFinal[j][6] += 1  print(tabulate(TabelaFinal)) |

|  |
| --- |
| # -\*- coding: utf-8 -\*-  """Tabela\_valores\_nulos.ipynb  Automatically generated by Colaboratory.  Original file is located at  https://colab.research.google.com/drive/1SbfAskd5v\_zCA\_EF3f4ExweSAw2d35R7  """  # informações que o usuário vai fornecer  import pandas as pd  import numpy as np  import datetime as dt  # import qadapo as vio  # importação do log de eventos - planilha d excel  log = pd.read\_excel('ConsultarValoresNulos.xlsx')  # nome da coluna caso  ColunaCaso = 'Caso'  # nome da coluna de atividades  ColunaAtividade = 'Atividade'  # nome da coluna caso  ColunaAtributo = 'Atributo'  # representacao do valor de nulo  nulos = '-'  # funcao para calcular a porcentagem  def porc(a,b):  porcentagem = a/b\*100 # "//" = 10//3 = 3.  return porcentagem  #LOG  valoresNulos = 0  total = 0  for i, linha in log.iterrows(): # dentro desse laço cada linha da tabela LOG está na variável LINHA  for coluna in log.columns: # dentro desse laço cada coluna esta na variavel COLUNA  total+=1 #incrementando o total de valores  if(linha[coluna]==nulos): #caso o valor seja NULO  valoresNulos+=1 #incremento o total de valores nulos  print("Análise do log",'\n')  print("Nulos\tTotal\tPorcentagem")  print(valoresNulos,'\t',total,'\t',porc(valoresNulos,total))  #EVENTO  print("Análise por evento",'\n')  valoresNulos = 0  total = 0  for i, linha in log.iterrows(): # dentro desse laço cada linha da tabela LOG está na variável LINHA  total+=1 # conta o total  for coluna in log.columns: # dentro desse laço cada coluna esta na variavel COLUNA  if(linha[coluna]==nulos): #caso o valor seja NULO  valoresNulos+=1 #incremento o total de valores nulos  break  print("Análise por evento",'\n')  print("Nulos\tTotal\tPorcentagem")  print(valoresNulos,'\t',total,'\t',porc(valoresNulos,total))  #ATIVIDADES  print("Análise por atividade",'\n')  listaAtividades = log[ColunaAtividade].unique() # serve para pegar a lista de atividades  for a in listaAtividades: # percorrer a lista de atividades, escolhida acima  valoresNulos = 0  total = 0  logFiltrado\_atividade = log[log[ColunaAtividade] == a ] # serve para filtrar o log da coluna de atividades  for i, linha in logFiltrado\_atividade.iterrows(): # dentro desse laço cada linha da tabela LOG está na variável LINHA  total+=1 # conta o total  for coluna in logFiltrado\_atividade.columns: # dentro desse laço cada coluna esta na variavel COLUNA    if(linha[coluna] == nulos): #caso o valor seja NULO  valoresNulos+=1 #incremento o total de valores nulos  break    print(a)  print("Nulos\tTotal\tPorcentagem")  print(valoresNulos,'\t',total,'\t',porc(valoresNulos,total),'\n')  #CASO  print("Análise por caso",'\n')  listaCaso = log[ColunaCaso].unique() # serve para pegar a lista de casos  for a in listaCaso: # percorrer a lista de casos, escolhida acima  valoresNulos = 0  total = 0  logFiltrado\_caso = log[log[ColunaCaso] == a ] # serve para filtrar o log da coluna de casos  for i, linha in logFiltrado\_caso.iterrows(): # dentro desse laço cada linha da tabela LOG está na variável LINHA  total+=1 # conta o total  for coluna in logFiltrado\_caso.columns: # dentro desse laço cada coluna esta na variavel COLUNA    if(linha[coluna] == nulos): #caso o valor seja NULO  valoresNulos+=1 # incremento o total de valores nulos  break    print(a)  print("Nulos\tTotal\tPorcentagem")  print(valoresNulos,'\t',total,'\t',porc(valoresNulos,total),'\n') |

|  |
| --- |
| # -\*- coding: utf-8 -\*-  """Tabela Ordem.ipynb  Automatically generated by Colaboratory.  Original file is located at  https://colab.research.google.com/drive/1IKA2CrWcXBEQ2Gt7ALqVJGYYvtX1OqAz  """  # informações que o usuário vai fornecer  import pandas as pd  import numpy as np  import datetime as dt  # import qadapo as vio  # importação do log de eventos - planilha d excel  log = pd.read\_excel('ConsultarOrdem.xlsx')  # nome da coluna caso  ColunaCaso = 'Caso'  # nome da coluna de atividades  ColunaAtividade = 'Atividade'  # nome da coluna do timestamp  ColunaTimestamp = 'Timestamp'  # informações da regra  primeiraAtv = 'Senha'  segundaAtv = 'Triagem'  # funcao para calcular a porcentagem  def porc(a,b):  porcentagem = a/b\*100 # "//" = 10//3 = 3.  return porcentagem  #CASO  print("Análise por caso",'\n')  listaCaso = log[ColunaCaso].unique() # serve para pegar a lista de casos  listaAtividades = log[ColunaAtividade].unique() # serve para pegar a lista de atividades  quantidadeViolacoes = 0  totalCasos = 0  for a in listaCaso: # percorrer a lista de casos, escolhida acima  totalCasos+=1  marcacao = False  print("\nCaso: ",a)  logFiltrado\_caso = log[log[ColunaCaso] == a ] # serve para filtrar o log da coluna de casos  logFiltrado\_ordenado = logFiltrado\_caso.sort\_values(ColunaTimestamp) # ordenar os valores da coluna do timestamp  for i in logFiltrado\_ordenado[ColunaAtividade]: # percorrer a lista de atividades daquele caso  if i == primeiraAtv: # ocorrencia da primeira atividade  marcacao = True  print(i)  if i == segundaAtv: # ocorrencia da primeira atividade  if marcacao == False: #nao ocorreu a primeira atividade  quantidadeViolacoes+=1  print('\n',"QuantidadeViolacoes\tTotalCasos\tPorcentagem")  print('\t',quantidadeViolacoes,'\t\t',totalCasos,'\t\t'  ,porc(quantidadeViolacoes,totalCasos),'\n') |