CEFET/RJ
DIRETORIA DE PESQUISA E PÓS-GRADUAÇÃO

UMA ANÁLISE DO USO DE LOTES CONFIGURÁVEIS NA DETECÇÃO DE EVENTOS EM SÉRIES TEMPORAIS EM *STREAMING*

Janio de Souza Lima

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Ciência da Computação, Centro Federal de Educação Tecnológica Celso Suckow da Fonseca CEFET/RJ, como parte dos requisitos necessários à obtenção do grau de mestre.

Orientadores:
Eduardo Soares Ogasawara
Rafaelli de Carvalho Coutinho

Rio de Janeiro,

22 de Janeiro de 2024

Uma Análise do Uso de Lotes Configuráveis na Detecção de Eventos em Séries Temporais em *Streaming*

Dissertação de Mestrado em Ciência da Computação, Centro Federal de Educação Tecnológica Celso Suckow da Fonseca, CEFET/RJ.

Janio de Souza Lima

Aprovada por:

Presidente, Professor D.Sc. Eduardo Soares Ogasawara(CEFET/RJ)

Professora D.Sc. Rafaelli de Carvalho Coutinho(CEFET/RJ)

Documento assinado digitalmente
gov.br
EDUARDO BEZERRA DA SILVA
Data: 11/12/2023 13:17:28-0300
Verifique em https://validar.iti.gov.br

Professor D.Sc. Eduardo Bezerra da Silva (CEFET/RJ)

Professor D.Sc. João Eduardo Ferreira (IME/USP)

Rio de Janeiro,

7 de Dezembro de 2023

USPAssina - Autenticação digital de documentos da USP

Registro de assinatura(s) eletrônica(s)

Este documento foi assinado de forma eletrônica pelos seguintes participantes e sua autenticidade pode ser verificada através do código QEPH-CJLF-11M5-TUI8 no seguinte link: https://portalservicos.usp.br/iddigital/QEPH-CJLF-11M5-TUI8

João Eduardo Ferreira

Nº USP: 827412

Data: 13/12/2023 15:19

Ficha catalográfica elaborada pela Biblioteca Central do CEFET/RJ

L732 Lima, Janio de Souza
Uma análise do uso de lotes configuráveis na detecção de eventos em séries temporais em streaming / Janio de Souza Lima. — 2024.
86f. : il. color. , enc.

Dissertação (Mestrado) Centro Federal de Educação Tecnológica Celso Suckow da Fonseca, 2024.
Bibliografia : f. 82-86
Orientador: Eduardo Soares Ogasawara
Coorientadora: Rafaelli de Carvalho Coutinho

1. Análise de séries temporais. 2. Análise de séries temporais - Anomalias. 3. Tecnologia streaming (Telecomunicação). I. Ogasawara, Eduardo Soares (Orient.). II. Coutinho, Rafaelli de Carvalho. (Coorient.). III. Título.

CDD 006.31

Elaborada pela bibliotecária Tania Mello – CRB/7 nº 5507/04

DEDICATÓRIA

A minha família, especialmente minha filha e minha esposa pelo apoio e compreensão nos momentos mais difícies para conciliar as diferentes responsabilidades educacionais, profissionais e familiares. A meus pais pela maior herança que me deixaram que foi o amor pela educação e a valorização de seu poder para mudar nossas vidas, independentemente de nossa origem.

AGRADECIMENTOS

Agradeço aos colegas do Grupo de Pesquisa *Data Analytics LAB* (DAL) Rebecca Salles, Luciana Escobar, Cristiane Géa, Heraldo Borges, Laís Baroni dentre outros pelos trabalhos de pesquisa conjuntos e contribuições para o aprimoramento profissional, assim como ao Prof. D.Sc Eduardo Bezerra, Prof. D.Sc Fábio Porto e Profa. PhD Esther Pacciti. Especialmente agradeço às orientações, correções e contínuo apoio no melhor direcionamento da pesquisa a meu orientador Prof. D.Sc. Eduardo Ogasawara e co-orientadora Profa. D.Sc. Rafaelli Coutinho.

Agradeço as agências de fomento CAPES, CNPq e à FAPERJ pelo apoio parcial deste trabalho.

Por fim, agradeço à Petrobras por meio de meus gestores imediatos pelo apoio para conciliar obrigações profissionais da empresa àquelas alusivas ao estudo e pesquisa.

RESUMO

Uma Análise do Uso de Lotes Configuráveis na Detecção de Eventos em Séries Temporais em *Streaming*

Janio de Souza Lima

Orientadores:
Eduardo Soares Ogasawara
Rafaelli de Carvalho Coutinho

Resumo da Dissertação submetida ao Programa de Pós-graduação em Ciência da Computação do Centro Federal de Educação Tecnológica Celso Suckow da Fonseca CEFET/RJ como parte dos requisitos necessários à obtenção do grau de mestre.

Detecção de eventos em séries temporais se refere à identificação de pontos que se diferenciam do comportamento esperado. Em cenários de alta conectividade, gêmeos digitais e tráfego de dados em nuvem observa-se o aumento da velocidade e do volume de geração dos dados de séries em *streaming*. Assim, a detecção de eventos é fundamental para tomada de decisões para correção ou prevenção de situações indesejadas. Divide-se a detecção de eventos em *online* e *offline*. Na detecção *online* o acesso aos dados é em tempo real, enquanto na detecção *offline* há acesso prévio ao conjunto de dados completo. A detecção *offline* não permite monitoramento dos dados enquanto eles são gerados. Contudo, a detecção a cada novo ponto na série pode trazer custos computacionais que reduzem a aplicabilidade dos métodos. Contudo, nota-se que, ainda que o processamento em lote possa ser associado ao processamento *offline*, mesmo o processamento de dados em *streaming* pode ocorrer por meio do fluxo de dados em pequenos lotes. Ainda há escassez de trabalhos com ferramentas para integração e avaliação de métodos voltados para o *streaming*. Mesmo em trabalhos existentes, não se identificam formas de analisar o comportamento dos métodos ao longo do *streaming*. A especificidade dos métodos existentes, a necessidade de equilíbrio entre o custo da detecção *online* e acurácia da detecção *offline* permitem levantar a questão: o uso de lotes configuráveis na detecção de eventos em séries em *streaming* resulta em detecção precoce e redução do custo computacional? Outras questões relevantes são: É possível avaliar o tempo entre a leitura de uma observação na série e sua detecção como evento? É possível avaliar o comportamento dos métodos ao longo do *streaming*? Para explorar as lacunas existentes na literatura, o presente trabalho propõe uma análise do uso de lotes configuráveis na detecção de eventos em séries temporais em *streaming*, avaliando seus impactos no equilibrio entre a acurácia e a detecção precoce de eventos. Além disso, o trabalho apresenta o *framework Nexus* para integração de métodos de detecção de eventos em *streaming* e métricas para avaliação de atraso na detecção e do comportamento dos métodos ao longo do *streaming*.

Palavras-chave:
Séries temporais, anomalias, eventos, *streaming*

Rio de Janeiro,

22 de Janeiro de 2024

ABSTRACT

Uma Análise do Uso de Lotes Configuráveis na Detecção de Eventos em Séries Temporais em *Streaming*

Janio de Souza Lima

Advisors:
Eduardo Soares Ogasawara
Rafaelli de Carvalho Coutinho

Abstract of dissertation submitted to Programa de Pós-graduação em Ciência da Computação - Centro Federal de Educação Tecnológica Celso Suckow da Fonseca CEFET/RJ as partial fulfillment of the requirements for the degree of master.

Time series event detection refers to identifying points that differ from expected behavior. In scenarios of high connectivity, digital twins, and cloud data traffic, an increase in the speed and volume of series data generation in *streaming* is observed. Thus, event detection is essential for making decisions to correct or prevent unwanted situations. Event detection is divided into *online* and *offline*. In *online* detection, data access is in real-time, while in *offline* detection there is prior access to the complete data set. *Offline* detection does not allow monitoring of data while it is generated. However, detecting each new point in the series can bring computational costs that reduce the applicability of the methods. However, it should be noted that, although batch processing can be associated with *offline* processing, even data processing in *streaming* can occur through data flow in small batches. There is still a lack of work on tools for integrating and evaluating methods aimed at *streaming*. Even in existing works, no ways of analyzing the behavior of methods throughout *streaming* have been identified. The specificity of existing methods, the need to balance between the cost of *online* detection and the accuracy of *offline* detection allow us to raise the question: the use of configurable batches in the event detection in series in *streaming* results in early detection and reduced computational cost? Other relevant questions are: Is it possible to evaluate the time between reading an observation in the series and its detection as an event? Is it possible to evaluate the behavior of methods throughout *streaming*? To explore the gaps in the literature, this work proposes an analysis of the use of configurable batches in event detection in time series in *streaming*, evaluating their impacts on the balance between accuracy and early detection of events. Furthermore, the work presents the *Nexus* framework for integrating event detection methods into *streaming* and metrics for evaluating delay in detection and the behavior of methods throughout *streaming*.

Key-words:
Time series, anomalies, events, streaming

Rio de Janeiro,

22 de Janeiro de 2024

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviações

AN	*Adaptive Normalization*	21, 33
ARIMA	*Autoregressive Integrated Moving Average*	25, 32, 34, 52, 57, 58, 59, 64, 71, 77
AWS	*Amazon Web Services*	32
CAD	*Conformal anomaly detection*	31
CF	*ChangeFinder*	31, 52, 53, 57, 58, 64, 67, 69, 71, 77
CNN	*Convolutional neural network*	25, 32, 33
CPU	*Central processing unit*	28, 34, 35, 40, 52
CRAN	*Comprehensive R Archive Network*	79
DP	Probabilidade De Detecção, Do Inglês *Detection probability*	41
ETL	*Extract, transform and load*	55
EWMA	*Exponentially Weighted Moving Average*	27, 31, 32, 33
FBIAD	*Forward and Backward Inertial Anomaly Detector*	27, 52, 53, 57, 58, 59, 64, 66, 69, 71, 77, 78
FN	Falsos Negativos	27, 28, 35
FP	Falsos Positivos	27, 28, 35
GARCH	*Generalized Autoregressive Conditional Heteroskedasticity*	25, 31, 52, 53, 74
HN	*Harbinger Nimbus*	79, 80
HW	*Holt-Winters*	31, 34
ICAD	*Inductive conformal anomaly detection*	31
IOT	*Internet of Things*	14, 76
KNN	*K-Nearest Neighbors*	31
KNN-CAD	*K-Nearest Neighbors-Conformal Anomaly Detector*	31, 33
LSTM	*Long Short Term Memory*	32, 52, 74
MAD	Microsoft Anomaly Detector	32, 33, 38, 39, 78, 80
MF	Filtro De Mediana	32
ML-AD	*Machine learning powered anomaly detection*	32, 38, 39
MTH	Memória Temporal Hierárquica	32, 34
NAB	*Numenta Anomaly Benchmark*	33, 34, 35, 36, 37, 55, 56, 67, 72, 73, 74, 75, 78
P2P	Ponto A Ponto	25

RARE	*Dataset for cloud-native memory anomalies*	35, 37, 55, 56, 67, 71, 72, 73, 78
SR	*Spectral residual*	33
TML	Tempo Médio Por Lote	62, 63, 64, 70, 75, 77
TVE	Teoria De Valores Extremos	33
UCI	*University of California, Irvine*	56, 68, 69, 71, 79
UCR	*University of California, Riverside*	36, 37, 56, 68, 69, 71, 72, 79
VN	Verdadeiros Negativos	27, 28, 35
VP	Verdadeiros Positivos	27, 28, 35

Capítulo I Introdução

Em cenários de alta conectividade, onipresença da internet, gêmeos digitais, avanço de dispositivos de Internet das coisas, do inglês *Internet of Things* (IoT), e tráfego de dados em plataformas de nuvem, é possível observar o aumento na velocidade de geração dos dados em *streaming* [Atzori et al., 2010; Ariyaluran Habeeb et al., 2019]. O *streaming* se refere a dados sendo gerados em um fluxo contínuo em tempo real, por exemplo, a partir do monitoramento de sensores. Diante do volume e da velocidade de geração dos dados, a rápida identificação de situações indesejadas é fundamental para tomada de decisões, permitindo ações corretivas ou preventivas [Ariyaluran Habeeb et al., 2019].

Uma série temporal é um conjunto de dados numéricos, organizado de forma sequencial e cronológica. Os dados gerados em *streaming* com esses atributos de tipo e organização caracterizam uma série temporal em *streaming*. Dentre os diversos problemas tratáveis com mineração de dados realizadas em séries temporais, temos a detecção de eventos [Esling and Agon, 2012]. A detecção de eventos consiste na busca de pontos ou sequências na série que não condizem com o esperado [Chandola et al., 2009].

A detecção de eventos abrange diversos cenários como identificação de fraudes, segurança cibernética, prevenção de perdas ou acidentes por meio de monitoramento de equipamentos na exploração de petróleo e mudanças inesperadas em séries temporais financeiras. É possível dividir a detecção de eventos quanto ao modo de acesso aos dados em *online* e *offline*. Denomina-se detecção *offline* ao ter acesso prévio à série completa, enquanto é denominada detecção *online* ao ter acesso aos dados em *streaming*, em tempo real à medida que cada observação da série é gerada [Truong et al., 2020].

Ao analisar séries em *streaming*, a tarefa de detecção de eventos traz dificuldades adicionais como a mudança no comportamento dos dados ao longo do tempo e atrasos nas detecções [Talagala et al., 2020; Hasani, 2020]. A possibilidade de mudança no comportamento dos dados gera necessidade de adaptação à medida que surgem novas observações dos dados em *streaming* . Além disso, é preciso monitorar os processos que geram os dados para detectar eventos em paralelo à sua execução, evitando atraso na detecção [Ariyaluran Habeeb et al., 2019; Talagala et al., 2020].

Trabalhos como de Talagala et al. [2020] e de Ariyaluran Habeeb et al. [2019] indicam dificuldade de aplicar métodos *offline* em séries em *streaming*. Contudo, não há evidências definitivas dessa

inaplicabilidade generalizada para qualquer método. Assim, mesmo ao estudar detecção *online* de eventos é importante analisar métodos criados inicialmente para detecção *offline* com uso amplamente difundido como nos trabalhos de Chandola et al. [2009], Salles et al. [2019], Salles et al. [2020] e Truong et al. [2020].

Na literatura há tanto trabalhos com objetivo de apresentar métodos para detecção em *streaming* como os propostos por Rettig et al. [2015]; Munir et al. [2019]; Zhang et al. [2020]; Ren et al. [2019] quanto pesquisas com foco na comparação de métodos como em Ahmad et al. [2017]; Talagala et al. [2020]; Salles et al. [2020]; Hasani [2020]; Belacel et al. [2022a]. Contudo, alguns desses trabalhos apenas apresentam os estudos e resultados da comparação, sem fornecer *frameworks* ou outras ferramentas para adição de novos métodos e formas de adaptá-los para o adequado funcionamento com séries em *streaming*.

Apesar de haver uma miríade de métodos voltados para a detecção de eventos [Truong et al., 2020; Ariyaluran Habeeb et al., 2019], ainda é possível observar uma escassez de trabalhos dedicados à experimentação abrangente de métodos de detecção de eventos em séries em *streaming*. Além disso, os trabalho de detecção de eventos em séries em *streaming* tem foco em métodos especializados por tipos de eventos e comportamento das séries. Assim, a escassez de trabalhos nessa área é acentuada no que se refere à experimentação de diferentes formas de aplicação dos métodos e diversidade dos conjuntos de dados [Ahmad et al., 2017; Lomio et al., 2020; Escobar et al., 2021; Wu and Keogh, 2021].

Outro aspecto a ser considerado é que apesar da detecção *offline* em lotes completos não atender às necessidades de velocidade de resposta, a detecção a cada novo ponto na série pode trazer custos computacionais que inviabilizem a aplicação prática de métodos [Talagala et al., 2020]. Ainda que o termo processamento em lote seja associado ao cenário *offline*, o processamento de dados em *streaming* pode ocorrer por meio de pequenos lotes [Rettig et al., 2015; Hiraman et al., 2018]. Isso ocorre, por exemplo, com a ferramenta *Spark streaming* que usa o conceito de processamento em micro lotes, ou seja, lotes pequenos dos dados para serem processados [Rettig et al., 2015].

A especificidade dos métodos para determinados comportamentos das séries e a necessidade de equilibro entre a detecção precoce de eventos e o custo do processamento *online* permitem levantar a seguinte hipótese: o uso de lotes configuráveis, que consigam analisar subsequências menores da série, na detecção de eventos em séries em *streaming* pode resultar em detecção precoce e redução do custo computacional do processamento? Além disso, é necessário que as detecções tenham boa acurácia e evite os impactos negativos da detecção tardia do processamento *offline* [Talagala et al., 2020; Ariyaluran Habeeb et al., 2019; Iwashita and Papa, 2019].

Ao se observar trabalhos de comparação de métodos cujos experimentos foram realizados em cenário *offline*, como Escobar et al. [2021], Gea et al. [2021] e Lima et al. [2022b,a], surgem questões

como: os resultados de métodos executados *offline* se mantém no cenário *online*? Métodos melhores em situações ou séries específicas continuam válidos em séries em *streaming*? Como o processamento das séries em *streaming* impacta o tempo de execução dos métodos? É possível empregar métodos com baixo custo computacional sem perder acurácia nas detecções?

A importância de monitorar continuamente séries em *streaming* para evitar atrasos nas detecções [Aminikhanghahi and Cook, 2017; Ariyaluran Habeeb et al., 2019; Talagala et al., 2020] permite levantar as seguintes questões adicionais: Como é possível avaliar o tempo decorrido entre o momento em que uma observação na série é lida e sua detecção como evento? É possível avaliar o comportamento dos métodos ao longo do *streaming*? Como informações sobre o comportamento dos métodos podem contribuir para identificar sua capacidade de adaptação às mudanças na série temporal em *streaming*?

A fim de explorar as lacunas existentes na literatura, o presente trabalho apresenta uma análise do uso de lotes configuráveis na detecção de eventos em séries temporais em *streaming*. Dessa forma, o trabalho contribui para avaliar os impactos para o equilíbrio entre a acurácia e a detecção precoce, bem como no custo computacional do uso de lotes em *streaming*. Além disso, este trabalho apresenta o *framework Nexus* para aplicação de lotes na implementação de novos métodos de detecção em *streaming*, bem como na adaptação de métodos já existentes.

O *Nexus* também fornece informações agregadas para análise do comportamento das detecções de eventos ao longo do *streaming* de dados. Todas as detecções efetuadas com o *framework* mantém informações históricas do funcionamento dos métodos ao longo dos lotes, como frequência em que um evento foi detectado, probabilidade de uma detecção corresponder a um evento real e atraso nas detecções. Esses são importantes diferenciais do *framework*, tendo em vista que os trabalhos existentes não fornecem informações detalhadas e estruturadas para análise desses aspectos.

Outro aspecto único do *Nexus* diz respeito às abordagens de tratamento de memória dos dados ao longo do *streaming*. Há uma dicotomia entre a detecção *offline* e *online* na literatura, mas tanto o uso de lotes deslizantes quanto o gerenciamento da memória de dados permite equilibrar o custo computacional e a capacidade de detecções acuradas por meio do *Nexus*.

Além disso, a realização da avaliação experimental abrangente contribui para: (i) avaliar o impacto de diferentes configurações de lotes deslizantes em séries em *streaming* como tamanho de lote e memória dos dados; (ii) comparação de diferentes métodos de detecção quanto à capacidade de detectar eventos corretamente e (iii) avaliação de tempo de execução e existência de atrasos nas detecções.

O *Nexus* foi testado em uma ampla diversidade de conjuntos de dados com séries relacionadas a tráfego de dados em nuvem, monitoramento de equipamentos de petróleo, dados ambientais e dados sintéticos. As séries avaliadas são oriundas de alguns dos principais *benchamarks* de detecção de

eventos, possibilitando um diagnóstico abrangente do funcionamento do *Nexus* e contribuem para generalização dos resultados obtidos. Como outra contribuição deste trabalho, foi disponibilizado um repositório público denominado *DAL Events Datasets* com a organização dos conjuntos de dados que pode ser acessado por meio de um pacote[1] na linguagem R.

Além dessa introdução, o trabalho apresenta no Capítulo II o referencial teórico. No Capítulo III são analisados trabalhos relacionados à detecção de eventos em séries temporais. No Capítulo IV é apresentada a metodologia para a análise do uso de lotes deslizantes na detecção de eventos em séries temporais em *streaming*. O Capítulo V apresenta a avaliação experimental e os resultados obtidos, enquanto o Capítulo VI tece considerações finais sobre o trabalho.

[1]Disponível em: https://github.com/cefet-rj-dal/event_datasets

Capítulo II Referencial Teórico

Este capítulo apresenta um referencial teórico sobre séries temporais e detecção de eventos. As primeiras duas seções discutem os conceitos básicos e peculiaridades das séries temporais. A Seção II.1 discorre sobre os conceitos e propriedades desse tipo de conjunto de dados. A Seção II.2 trata de tarefas transformação de dados aplicada a séries temporais. A Seção II.4 aborda séries temporais em *streaming* e seus impactos na detecção de eventos, enquanto a Seção II.5 discute métricas para avaliação da detecção de eventos.

II.1 Séries temporais

Uma série temporal é um conjunto de dados numéricos, organizados de forma sequencial e cronológica. A Equação II.1 representa uma série temporal, onde X é a série com n observações, x_1 é a observação mais antiga e x_n é a observação mais recente, ilustrada a seguir pela Figura II.1[1]. Essa representação refere-se a uma série com apenas uma variável, denominada série temporal **univariada**, enquanto uma série com diversas variáveis é denominada **multivariada** [Esling and Agon, 2012].

$$X = (x_1, ..., x_n), x_i \in R \tag{II.1}$$

Algumas características das séries temporais dificultam a aplicação de métodos estatísticos tradicionais ou requerem técnicas específicas de análises. Por exemplo, a assunção de independência e distribuição idêntica entre as observações adjacentes nos dados analisados feita por métodos estatísticos vai de encontro à autocorrelação entre as observações de uma série [Shumway and Stoffer, 2017]. Inclusive, é comum estudar o comportamento de uma série univariada em função de seus dados passados [Hanssens et al., 2003; Ogasawara et al., 2021].

A noção de estacionariedade refere-se a características de uma série que mantém regularidade de algumas medidas estatísticas, como média, variância e autocovariância, ao longo do tempo [Salles et al., 2019; Shumway and Stoffer, 2017]. Muitos métodos aplicados à análise de séries temporais assumem sua estacionariedade. Contudo, é muito comum que uma série temporal seja não estacionária, pois ao violar qualquer das restrições de uma série estacionária, já podemos considerá-la

[1]Dados disponíveis em: https://www.kaggle.com/gustavomodelli/forest-fires-in-brazil

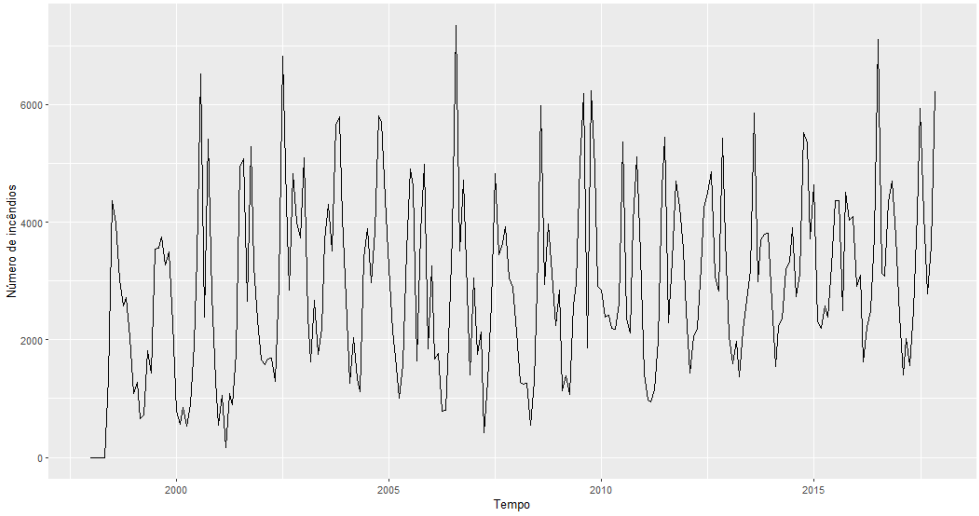


Figura II.1: Série temporal de queimadas em florestas no Brasil de 1998 a 2017: No eixo das ordenadas temos os valores das observações x_1 a x_n e no eixo das abscissas temos o tempo no qual as observações estão distribuídas.

não estacionária. A ocorrência de não estacionariedade da série pode indicar presença de eventos [Ogasawara et al., 2021].

Como efeito da não estacionariedade, o comportamento típico dos dados muda à medida que novas observações ocorrem com pontos de mudança ou alterações na sazonalidade e ciclos. Assim, modelos aprendidos baseados em observações passadas perdem a capacidade de explicação do comportamento futuro da série. Para resolver esse problema, os modelos na análise de séries não estacionárias precisam ser atualizados periodicamente [Talagala et al., 2020].

II.2 Transformação de dados aplicada a séries temporais

Esta seção apresenta algumas das principais tarefas de transformação de dados aplicadas a séries temporais. Tarefas de transformação geram versões ou representações da série temporal. As transformações são necessárias para tratar características que dificultam a aplicação de técnicas tradicionais em séries temporais, por exemplo, a não estacionariedade [Lin et al., 2003; Shumway and Stoffer, 2017].

Como parte das tarefas de pré-processamento de dados, há diversas técnicas para transformação de séries temporais como decomposição, normalização, suavização, extração de janelas deslizantes [Ogasawara et al., 2010; Shumway and Stoffer, 2017; Lin et al., 2003]. A seguir são apresentadas algumas dessas técnicas que fazem parte da preparação de dados para melhorar a qualidade dos dados antes da aplicação de tarefas de mineração mais complexas [Ogasawara et al., 2010; Han et al., 2012].

A decomposição de séries temporais visa identificar componentes latentes da série que normalmente não são observáveis. Os principais componentes de uma série são: tendência, ciclos,

sazonalidade e resíduos. A tendência indica a direção de subida ou queda dos valores das observações. Os ciclos são ocorrências de picos ou vales de maneira repetitiva ao longo do tempo. A sazonalidade é similar aos ciclos, mas refere-se à sua ocorrência repetitiva em cada ano. Os resíduos são os componentes restantes, relativos a variações não previsíveis [Shumway and Stoffer, 2017].

Para ilustrar o que são os componentes e a decomposição de séries temporais, pode-se analisar a Figura II.2, que retoma a série apresentada na Figura II.1 e aplica a decomposição. O primeiro gráfico da Figura II.2 representa a série original com dados de queimadas no Brasil no período de 1998 a 2017[2]. Os demais gráficos representam, respectivamente, os componentes latentes presentes na série: tendência, sazonalidade e resíduos ou ruído aleatório, que não podem ser observados diretamente na série original.

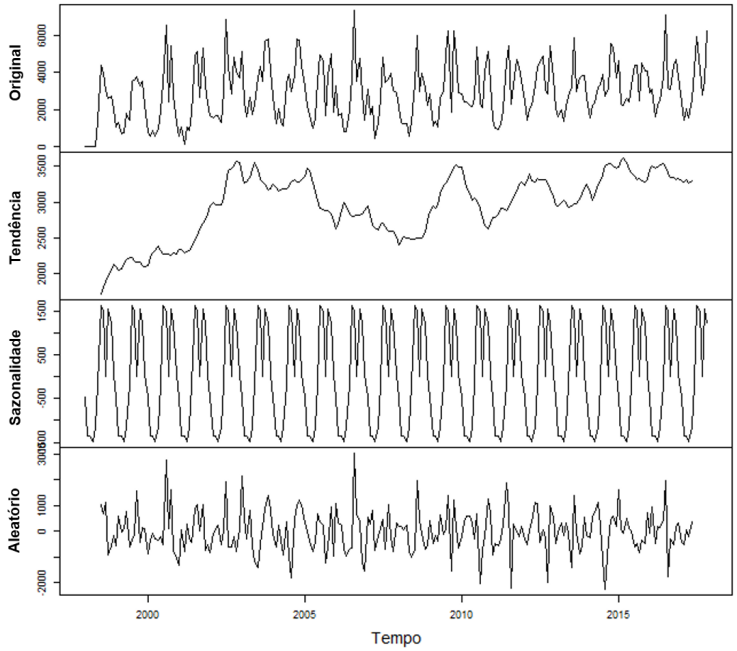


Figura II.2: Exemplo de decomposição de séries: série temporal de queimadas em florestas no Brasil 1998 a 2017 decomposta em tendência, sazonalidade e componente aleatório.

A decomposição apresentada na Figura II.2 é realizada removendo a tendência, por meio da Equação II.2, onde X_t é a série original e T_t é o componente de tendência. Em seguida é removida a sazonalidade, por intermédio da Equação II.3, onde S_t é o componente sazonal. Com isso, chega-se à série $\hat{X}_t$ apenas com resíduos ou ruído aleatório [Shumway and Stoffer, 2017].

$$\hat{X}_t = X_t - T_t \tag{II.2}$$

$$\hat{X}_t = \hat{X}_t - S_t \tag{II.3}$$

[2]Dados disponíveis em: https://www.kaggle.com/gustavomodelli/forest-fires-in-brazil

A decomposição de uma série permite entender melhor o seu comportamento pelo estudo dos componentes latentes, mesmo em uma análise exploratória inicial. Outra aplicação comum da decomposição é a aplicação de técnicas de detecção de eventos na série diferenciada (série obtida pela remoção de um ou mais componentes), lidando com a não estacionariedade relacionada a algum dos componentes. Um problema de técnicas que usam a decomposição é o alto custo de processamento [Marik and Bohac, 2019].

Outras técnicas relevantes de transformação de dados em séries temporais são a normalização e suavização de séries temporais que, assim como na transformação de dados em geral [Han et al., 2012], geram representações simbólicas da série que podem tornar as tarefas de análise mais eficientes. Como forma de suavização, é comum o uso de médias móveis. As médias móveis, muito usadas em finanças e econometria, reduzem o efeito de ruídos e auxiliam na identificação de mudanças no comportamento da série [Ogasawara et al., 2010].

A normalização de dados consiste em mapear valores equivalentes para cada observação do conjunto de dados em uma escala diferente, por exemplo, em um intervalo definido [Han et al., 2012]. Essas representações auxiliam no equilíbrio do peso entre atributos diferentes em um conjunto de dados e facilita tarefas como agrupamento e classificação. Os métodos *min-max*, *z-score* e escala decimal estão entre os mais comuns para normalização [Ogasawara et al., 2010; Han et al., 2012].

O método *min-max* gera uma representação em que cada observação está em um intervalo definido de valores mínimo e máximo. Contudo, se os valores da série não forem conhecidos, a definição desses limites pode gerar erros. Isto pode ocorrer pela concentração dos dados normalizados em uma faixa indevida [Han et al., 2012].

A normalização pelo *z-score* gera um novo conjunto de valores denominados valores padronizados ou valores z. Para tanto, dada uma série X com desvio padrão σ e média μ, obtém-se para cada x_i o valor z pela Equação II.4. Como usa os valores de média e desvio padrão dos dados como referência, pode ser afetada por séries não estacionárias nas quais essas medidas variam ao longo do tempo.

$$z = \frac{x_i - \mu}{\sigma} \tag{II.4}$$

Por sua vez, a normalização pela escala decimal consiste no deslocamento do ponto decimal dos valores da série segundo o valor máximo existente [Han et al., 2012]. Outra técnica que pode ser aplicada nessa tarefa é a *Adaptive Normalization* (AN) [Ogasawara et al., 2010], que consiste em transformar a série não estacionária em estacionária usando a técnica de janelas deslizantes, remover valores discrepantes e somente depois desses passos normalizar a série.

Para analisar propriedades locais em uma série X, podem ser extraídas subsequências de tamanho p, formalmente definidas na Equação II.5. As subsequências extraídas são valores con-

tínuos na série no intervalo definido na Equação II.5 e podem ser usadas tanto como tarefa de pré-processamento como para busca de padrões, técnica de análise de tendência ou discretização de dados [Fu, 2011]. Uma técnica comum de subsequências é a extração de janelas deslizantes.

$$seq_{p,i}(X) =< X_i, X_{i+1}, ..., x_{i+(p-1)} >, |seq_{p,i}(X)| = p, 1 \leq i \leq |X| - p \tag{II.5}$$

A técnica de janelas deslizantes consiste em extrair subsequências que percorrem toda a série. Para isso gera-se um conjunto de subsequências de mesmo tamanho, podendo ser representada como uma matriz A contendo todas as subsequências da série X, como definida na Equação II.6. A extração de janelas deslizantes serve para trabalhar com análises das propriedades locais de cada janela [Lin et al., 2003].

$$sw_p(X) = A, \forall a_i \in A, a_i = seq_{p,i}(X) \tag{II.6}$$

Também é possível analisar uma representação da série original completa, deslizando através de suas subsequências ou analisando cada janela como um termo defasado da série [Lin et al., 2003; Ogasawara et al., 2010]. Também utilizam-se as janelas deslizantes para realizar o agrupamento de subsequências. O agrupamento de subsequencias trata cada janela como uma série individual e busca mensurar as similaridades entre elas para identificar grupo de janelas semelhantes, mesmo que não sejam contínuas [Lin et al., 2003].

II.3 Eventos em séries temporais

As tarefas de mineração de dados representam uma organização didática dos temas, mas podem ocorrer de maneira conjunta [Esling and Agon, 2012]. Além disso, normalmente um tipo de tarefa usa resultados de outra ou lhe é complementar. Nesta seção são discutidos conceitos relacionados à tarefa de detecção de eventos.

Ao pesquisar por eventos em séries temporais, alguns termos são encontrados na literatura de maneira intercambiável como eventos, anomalias, discrepâncias, valores atípicos. Alguns autores, como Chandola et al. [2009], simplesmente consideram alguns deles como sinônimos, como anomalias e valores atípicos. Por outro lado, às vezes eventos e anomalias são tratados como subcategorias um do outro, como citado por Marik and Bohac [2019], que se referem a anomalias como algo mais genéricos e eventos como um padrão mais complexo encontrado na série.

Neste trabalho é usado o termo evento de forma mais geral para aludir a pontos em uma série temporal com significado especial. Apesar de identificados como pontos específicos, tem relação com intervalos da série temporal ao seu redor. Os eventos podem referir-se, por exemplo, a anomalias e pontos de mudança [Gensler and Sick, 2018].

A detecção de eventos refere-se à detecção automática de pontos com significado especial em uma série temporal. Uma forma de dividir a análise das séries para detecção de eventos é quanto ao modo de acesso aos dados em *online* e *offline*. Quando o acesso é em tempo real, à medida que são geradas novas observações, a análise é denominada detecção *online*, e denomina-se detecção *offline* ao ter acesso prévio ao conjunto de dados completo [Truong et al., 2020].

A Figura II.3 apresenta uma taxonomia para detecção *online* de eventos em séries temporais, proposta por [Ogasawara et al., 2021] baseada numa análise da literatura sobre o tema e inspirada em [Ariyaluran Habeeb et al., 2019]. A taxonomia, adaptada como contribuição deste trabalho, divide o estudo de detecção *online* de eventos em séries nas categorias: (i) tipos e (ii) granularidade de eventos, (iii) forma de aprendizado, (iv) estratégia, (v) abordagens e (vi) base de detecção. O restante desta seção trata sobre alguns dos aspectos relevantes dessa taxonomia e da literatura em geral para o estudo de eventos em séries temporais.

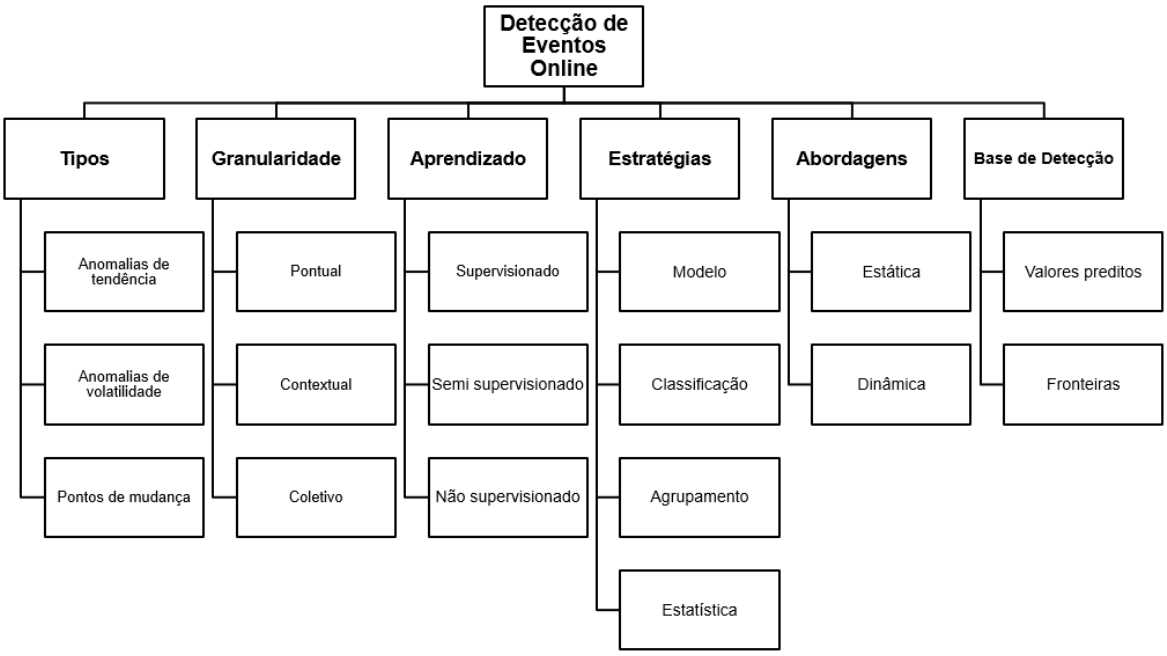


Figura II.3: Taxonomia de detecção de eventos *online*. Adaptada de Ogasawara et al. [2021]

A adaptação da taxonomia ilustrada na Figura II.3 refere-se à base de detecção que pode ser detecção baseada em divergência de valores preditos ou fronteiras. Na detecção baseada em valores preditos, os métodos possuem uma etapa para predição dos próximos valores da série e cada nova observação no *streaming* é comparada com a predição, sendo marcados como eventos quando há divergência entre o real e o predito. A base em fronteiras consiste em definir limites inferiores e superiores além dos quais cada nova observação recebida no *streaming* é marcada como um evento. Esse comportamento pode ser observado ao estudar métodos em trabalhos como Ahmad et al. [2017]; Munir et al. [2019]; Talagala et al. [2020]; Zhang et al. [2020]; Ren et al. [2019].

Um tipo comum de eventos são as anomalias, que se referem a observações que não condizem

com o comportamento esperado do conjunto de dados [Chandola et al., 2009]. Quanto aos tipos, as anomalias podem ser de tendência ou de volatilidade. As anomalias de tendência ocorrem quando um evento escapa da tendência esperada da série, enquanto as anomalias de volatilidade ocorrem quando há uma grande mudança na volatilidade da série analisada [Ogasawara et al., 2021].

Pontos de mudança são eventos que indicam uma mudança repentina na série temporal. Sua identificação é relevante porque após o ponto de mudança é raro identificar a sequência de observações em uma série temporal com base em seu comportamento antes da mudança. O principal interesse na identificação de pontos de mudança é em mudanças abruptas, pois mudanças mais suaves são naturais nas séries [Takeuchi and Yamanishi, 2006].

Conforme ilustrado pela segunda categoria da Figura II.3, quanto à sua granularidade, os eventos podem ser divididos em pontuais, coletivos ou contextuais [Chandola et al., 2009; Han et al., 2012]. Os pontuais ocorrem quando um ponto individual já é considerado um evento em relação aos demais dados. Os coletivos ocorrem quando um ponto individualmente não é considerado um evento, mas sua ocorrência em conjunto com outros pontos não condiz com o comportamento do conjunto de dados. Quando a discrepância é condicionada a determinado contexto, relacionado à estrutura do conjunto de dados, temos um evento contextual [Chandola et al., 2009; Han et al., 2012].

Dependendo da disponibilidade de rótulos para identificar a ocorrência de eventos, as técnicas de detecção podem ser classificadas como supervisionada, semi supervisionada ou não supervisionada [Chandola et al., 2009; Ariyaluran Habeeb et al., 2019; Han et al., 2012]. As técnicas supervisionadas ocorrem quando há disponibilidade de rótulos para treinamento tanto para os casos normais quanto para anomalias, enquanto quando há apenas rótulos para os casos normais têm-se técnicas semi supervisionadas. Quando não há qualquer rótulo tem-se a detecção não supervisionada.

Outra importante classificação das técnicas de detecção de eventos é quanto à estratégia empregada. As estratégias das técnicas podem ser classificação, agrupamento, análise de desvio de modelos ou técnicas estatísticas [Chandola et al., 2009]. Assim como outros aspectos da taxonomia, as estratégias não funcionam de maneira isolada, mas são empregadas em conjunto com as formas de aprendizado e base de detecção.

As técnicas baseadas em classificação e agrupamento tratam a detecção como tarefas de classificação e agrupamento, respectivamente, onde se busca atribuir uma classe ou grupo para cada ponto na série [Esling and Agon, 2012]. A análise de desvio de modelo consiste no treinamento prévio de um modelo e posterior comparação das novas observações para identificar como eventos aquelas que desviam do modelo treinado. Por sua vez, técnicas estatísticas trabalham com a análise da distribuição dos dados e marcação de eventos para observações que não estão em conformidade com a distribuição identificada para a série [Ogasawara et al., 2021].

Os métodos podem usar estratégicas específicas, como métodos clássicos usados em economia

como *Autoregressive Integrated Moving Average* (ARIMA) [Shumway and Stoffer, 2017] e *Generalized Autoregressive Conditional Heteroskedasticity* (GARCH) [Carmona, 2013] que empregam técnicas estatísticas. Outros podem inclusive usar um híbrido de técnicas, como *FuseAD* que combina ARIMA e aprendizado de máquina baseados em modelos como *convolutional neural network* (CNN) [Shumway and Stoffer, 2017].

Há duas abordagens principais para detecção *online* de eventos em séries em *streaming*: modelos estáticos e modelos dinâmicos, conforme se observa na taxonomia da Figura II.3. Nos modelos estáticos o método treina um modelo em grandes volumes de dados e posteriormente o aplica no *streaming* de dados, marcando cada observação como evento ou não à medida que há desvios em relação ao modelo treinado. Nos modelos dinâmicos, o treinamento é feito apenas como aquecimento para iniciar o modelo e realizar as detecções, mas à medida que novos dados são recebidos via *streaming*, o modelo é retreinado continuamente de maneira incremental [Ariyaluran Habeeb et al., 2019].

II.4 Séries temporais em *streaming*

O *streaming* de dados está relacionado ao tráfego e processamento de dados em tempo real, antes de seu armazenamento. Esse tipo de processamento de dados é muito comum em diferentes cenários como redes sociais, monitoramento de sensores, plataformas de *streaming* de vídeo e monitoramento de valores gerados por sensores em equipamentos industriais e de séries financeiras, como preço de ações. Uma forma comum para o processamento de dados em *streaming* é conhecida como sistemas de mensageria, em que os dados trafegam como mensagens [Hiraman et al., 2018].

Há dois padrões principais para sistemas de dados em *streaming*: (i) ponto a ponto (P2P) e (ii) publicar/subscrever PUB/SUB. No padrão, P2P as aplicações conectam-se entre si, enviando e recebendo dados trafegados em uma fila. No segundo padrão há publicadores de conteúdo e consumidores desse conteúdo, sem necessidade de conexão direta entre as aplicações [Hiraman et al., 2018].

No contexto de processamento de dados em *streaming* é importante citar a existência de plataformas especializadas nessa tarefa. Ariyaluran Habeeb et al. [2019] citam *Apache Kafka* e *Spark streaming* como exemplos de plataformas específicas para processamento de *streaming* de dados. Essas plataformas seguem o padrão PUB/SUB para troca de dados em tempo real e tratamento de grandes volumes de dados [Hiraman et al., 2018].

Além do tráfego de dados, as plataformas de *streaming* apresentam diversas possibilidades de análise desses dados na forma de séries temporais, tais como preços de ações, tráfego de rede, consumo de recursos da nuvem, e vazamentos em dutos de petróleo. Em resumo, qualquer dado de natureza numérica trafegando em *streaming* que possa ser organizado de maneira cronológica e

regular pode ser analisado como uma série temporal em *streaming*. Há diversas tarefas na análise dessas séries temporais, como agregações, predições, detecção de fraudes e detecção de eventos [Carter and Streilein, 2012].

Na detecção de eventos *offline* os métodos precisam do acesso aos dados completos da série analisada. Por outro lado, na *online* os métodos analisam a série à medida que novas observações são geradas [Truong et al., 2020]. Assim, ainda que seja possível armazenar dados coletados de uma série temporal para sua posterior análise *offline* como um lote completo, essa tarefa contribui mais para entender fenômenos passados.

Entretanto, ao se trabalhar com eventos em séries em *streaming* o foco é no uso das informações em tempo real ou próximo ao tempo real, por isso se faz necessária a detecção *online* [Rettig et al., 2015; Ariyaluran Habeeb et al., 2019]. Isto é relevante para que a geração dos dados e o seu monitoramento ocorra de maneira paralela. Portanto, a detecção de eventos à medida que os dados são gerados permite ações corretivas para evitar efeitos indesejados dos eventos.

A detecção em lotes completos (*offline*) não atende às necessidades de velocidade de resposta e monitoramento dos dados enquanto eles são gerados. Contudo, a detecção a cada novo ponto na série pode trazer custos computacionais que reduzem a aplicação prática de métodos [Talagala et al., 2020]. Nesse contexto, deve-se notar que, ainda que o termo processamento em lote seja normalmente associado ao processamento *offline*, mesmo o processamento de dados em *streaming* pode ocorrer por meio do envio e recepção de dados individuais ou de pequenos lotes [Truong et al., 2020; Rettig et al., 2015; Hiraman et al., 2018].

O uso de lotes ocorre, por exemplo, com a ferramenta *Spark streaming* que usa micro lotes, ou seja, lotes pequenos dos dados para serem processados [Rettig et al., 2015]. Uma intuição similar para divisão da série em lotes foi experimentada para o processamento em *streaming* para identificação de mudanças de conceito por Iwashita and Papa [2019]. Zeileis et al. [2002] apresentam um pacote na linguagem R denominado *strucchange* com uma aplicação similar, adaptada do conceito de janelas deslizantes, para detectar pontos de mudança em *streaming*.

Percebe-se em abordagens como a do *Spark* [Rettig et al., 2015] e de Iwashita and Papa [2019] tentativas de equilíbrio entre a redução de custo computacional e alta velocidade de resposta. Esse tipo de abordagem pode indicar um caminho para o meio-termo no enfrentamento do desafio de não precisar processar a série como um lote completo nem precisar realizar todas as etapas de aprendizado e detecção a cada novo ponto emergindo da série.

O uso de lotes na detecção em *streaming* não é presente em trabalhos de detecção de eventos. Contudo, além do trabalho de Iwashita and Papa [2019] que aplica o conceito à detecção de mudanças de conceito em séries. Há também uma adaptação do uso de janelas deslizantes em tarefa de detecção de pontos de mudança desenvolvida por Zeileis et al. [2002].

As janelas deslizantes, representarem um conceito similar à análise da série em lotes, fornecendo uma intuição importante sobre a possibilidade de usar subsequências para analisar as séries de maneira progressiva. Métodos como normalização adaptativa [Ogasawara et al., 2010], *Exponentially Weighted Moving Average* (EWMA) [Carter and Streilein, 2012] e *Forward and Backward Inertial Anomaly Detector* (FBIAD) [Lima et al., 2022a] usam janelas deslizantes em etapas de pré-processamento isoladamente ou como etapas intermediárias da detecção de eventos.

A detecção *online* de eventos de série temporais em *streaming*, apesar de relevante para monitoramento dos dados, traz novos desafios como ausência de rótulos identificando os eventos reais, capacidade de adaptação para diferentes tipos de séries e aprendizado de parâmetros com os próprios dados [Carter and Streilein, 2012]. Além disso, os dados nas séries em *streaming* podem passar a maioria do tempo com comportamento normal, gerando conjuntos de dados desbalanceados quanto às classes normais e anômalas. Por isso, o monitoramento e a identificação de eventos podem ter alto consumo de recursos e ser ineficientes.

Ademais, o comportamento de série em *streaming*, como mudanças ao longo do tempo e presença de não estacionariedade, dificulta a aplicação de abordagens tradicionais usadas na detecção *offline* de eventos [Talagala et al., 2020; Ahmad et al., 2017]. O trabalho com lotes completos de dados, por exemplo, é incompatível com a necessidade de detecção precoce de eventos [Talagala et al., 2020]. Sem a detecção precoce em tempo real (a cada nova observação) ou próxima ao tempo real, a capacidade de resposta aos eventos fica prejudicada.

II.5 Avaliação de detecção de eventos

Outro aspecto importante na detecção de eventos é a avaliação dos métodos quanto à assertividade de seus resultados. Esta seção descreve como as métricas de classificação são adaptadas como forma de avaliar métodos de detecção de eventos em séries temporais. A Tabela II.1 apresenta a Matriz de Confusão com a tabulação dos acertos e erros de detecção e II.2 apresenta métricas comuns em tarefas de classificação como acurácia, precisão, revocação e F_1 [Han et al., 2012].

A Tabela II.1 ilustra um exemplo de uma matriz de confusão, na qual são distribuídos os valores da contagem de observações da série com sua classificação. Os valores da matriz de confusão indicam a contagem de: verdadeiros positivos (VP), acertos de existência de eventos; verdadeiros negativos (VN), acertos de inexistência de eventos; falsos positivos (FP) e falsos negativos (FN) [Han et al., 2012]. Esses valores são a base para o cálculo das principais métricas descritas na Tabela II.2 que permite o entendimento da capacidade de detecções corretas dos métodos analisados.

A matriz de confusão pode ser usada em classificações mais complexas em outros tipos de conjuntos de dados com várias classes, mas a matriz adaptada na Tabela II.1 é o tipo mais básico, usada normalmente para classificações em conjuntos de dados de duas classes [Han et al., 2012]. Por

Tabela II.1: Matriz de Confusão de Detecção de Eventos. Adaptada de Han et al. [2012]

		Detectados	
		Sim	Não
Eventos	Sim	VP	FN
Reais	Não	FP	VN

isso, esse tipo de matriz é adequada para análises dos eventos, pois facilita a análise simplificada de acertos (VP, VN) e erros (FP, FN). Sua interpretação é intuitiva, pois visualmente é possível verificar o desempenho de um método que é melhor à medida que há maior concentração de observações na diagonal formada pelos acertos (VP, VN).

Comumente, ao analisar o desempenho de métodos de detecção de eventos usam-se as métricas de classificação descritas na Tabela II.2 [Ren et al., 2019; Salles et al., 2020; Hasani, 2020]. O resultado da tarefa de detecção de eventos normalmente contém um vetor lógico (verdadeiro/falso) ou binário (1/0) que indica se cada observação da série é um evento. Assim, a partir da comparação dos resultados obtidos com os rótulos indicando os eventos reais, é possível gerar uma matriz de confusão.

Tabela II.2: Métricas para avaliação de acertos das detecções: obtidas a partir da matriz de confusão

Métrica	**Descrição**	**Fórmula**
Acurácia (AC)	Proporção de acertos em relação ao total de observações	$\frac{VP+VN}{VP+VN+FP+FN}$
Precisão (PR)	Medida de exatidão	$\frac{VP}{VP+FP}$
Revocação (RV)	Medida de completude ou taxa de verdadeiro positivo	$\frac{VP}{VP+FN}$
F_1	Medida que avalia o equilíbrio entre precisão e revocação	$2 \times \frac{PR \times RV}{PR+RV}$

As métricas apresentadas na Tabela II.2, por sua vez, permitem uma análise mais aprofundada do que a matriz de confusão. Com a análise da acurácia é possível ter uma visão geral dos acertos, ou seja, indica se determinado método consegue acertar tanto a existência de eventos quanto a inexistência. A precisão fornece informações sobre o nível de acertos dos eventos existentes, mas não consegue identificar se foram marcados muitos pontos como falsos negativos.

De maneira inversa à precisão, a revocação consegue avaliar se o método classificou o maior número de eventos possíveis, mas não captura excesso de falsos positivos. A precisão e revocação tendem a ter uma relação inversamente proporcional. Assim, a métrica F_1 permite avaliar se um método obteve resultados equilibrados, pois é obtida pelo cálculo da médica harmônica entre precisão e revocação, permitindo [Han et al., 2012].

Além das métricas calculadas a partir na matriz de confusão, é comum avaliar aspectos como tempo de execução, consumo de recursos como memória e *Central processing unit* (CPU) [Hasani,

2020]. Além disso, há trabalhos como Ahmad et al. [2017] e Salles et al. [2023b] que propõem novas métricas específicas para detecção de eventos. Na análise de trabalhos relacionados na próxima seção, a discussão sobre métricas é retomada para observar quais são comumente usadas em trabalhos com foco na comparação de métodos.

Neste capítulo foram explorados os principais conceitos relacionados às séries temporais, suas características básicas e tarefas de mineração de dados aplicadas. Também foram abordados aspectos relevantes da detecção de eventos, particularidades relacionadas ao contexto de séries em *streaming* e métricas para avaliação de métodos de detecção. Partindo desse conhecimento do referencial teórico, o próximo capítulo aborda os principais trabalhos relacionados à detecção de eventos em séries temporais.

Capítulo III Trabalhos Relacionados

A discussão dos trabalhos relacionados foi organizada neste capítulo para explorar tanto as especificidades dos métodos quanto as formas de comparação desses métodos. Para isso, trabalhos que propõem métodos de detecção de eventos em séries são explorados na Seção III.1 e trabalhos de comparação de métodos, métricas e conjuntos de dados são discutidos na Seção III.2. O capítulo é concluído com a Seção III.3 que sintetiza os achados da análise dos trabalhos, aborda lacunas identificadas e aspectos relevantes para a evolução da pesquisa.

O intuito da divisão da análise dos trabalhos relacionados dessa forma foi permitir analisar de maneira separada os métodos e os trabalhos de exploração mais abrangentes dessa área de pesquisa. Isso é relevante, tendo em vista que um dos objetivos desta pesquisa é avaliar diversos métodos e fornecer um panorama abrangente da detecção de eventos em séries temporais em *streaming*. Uma análise de trabalhos relacionados que abordasse apenas os métodos de detecção poderia deixar lacunas, como o entendimento da forma como outros pesquisadores selecionam conjuntos de dados e métricas de desempenho para comparação de métodos.

III.1 Métodos de detecção de eventos em séries em *streaming*

Para exploração da área de pesquisa sobre detecção de eventos *online* em séries temporais, inicialmente foram realizadas buscas de artigos indexados na base *Scopus*. Para seleção foi usada a *string* de busca *"time series" AND ("anomaly detection" OR "event detection") AND ("online" OR "streaming")* aplicada aos títulos, resumos e palavras-chave, obtendo-se 466 documentos na última consulta em abril de 2023. Para resultados relevantes foram explorados os documentos, eliminando-se artigos sem citações, exceto quando publicados nos últimos 12 meses. Foram analisados os resumos dos documentos mantidos no filtro para identificar associação ao tema estudado e aplicada a técnica de *snowballing*, resultando em 31 documentos.

Há diversos métodos desenvolvidos para detecção de eventos *offline*, contudo trabalhos como de Talagala et al. [2020]; Ariyaluran Habeeb et al. [2019] indicam dificuldade desses métodos lidarem com séries em *streaming*. Assim, esta seção foca em métodos voltados para detecção em tempo real. Contudo, segundo o melhor conhecimento dos autores, não há evidências definitivas da inaplicabilidade de métodos indicados inicialmente para detecção *offline* no cenário de *streaming*. Diante disto

e tendo em vista seu uso amplamente difundido para tipos de séries ou de eventos específicos que não devem ser desprezados, é importante também citá-los nesta seção [Salles et al., 2020; Truong et al., 2020; Salles et al., 2019; Chandola et al., 2009].

Alguns exemplos de métodos desenvolvidos inicialmente para detecção *offline* conforme o tipo de eventos são: (i) **Anomalias de volatilidade**: modelos econométricos como GARCH [Carmona, 2013]; (ii) **Anomalias de tendência**: métodos para identificar a tendência na série como *Holt-Winters* (HW) [Hasani, 2017]; (iii) **Pontos de mudança**: EWMA e *ChangeFinder* (CF) [Guralnik and Srivastava, 1999; Takeuchi and Yamanishi, 2006; Escobar et al., 2021].

Há também determinados métodos ou ferramentas com aplicação híbrida, ou seja, desenvolvidas tanto para detecção *offline* quanto *online*. Na análise dos trabalhos foram identificadas *Strucchange* [Zeileis et al., 2002] e *k-Nearest Neighbors-Conformal Anomaly Detector* (KNN-CAD) [Salles et al., 2020] com esse tipo de abordagem híbrida. Ambos são apresentados a seguir.

Zeileis et al. [2002] apresentaram *Strucchange*, um pacote na linguagem R que permite a avaliação de pontos de mudança por meio de testes estatísticos de mudança de estrutura na série temporal. Os testes disponíveis são *F test*, CUSUM e MOSUM que avaliam as mudanças baseado em modelos de regressão linear da série temporal. Apesar de não tratar de outros tipos de eventos, como anomalias, o *strucchange* permite tanto a avaliação *offline* quanto *online* na aplicação de testes em séries em *streaming* por meio de janelas deslizantes[1].

O método KNN-CAD identifica eventos do tipo anomalia em séries univariadas de maneira *offline* e *online*. O KNN-CAD combina mensuração de distância com análise de fronteiras, para isso usa o algoritmo *k-Nearest Neighbors* (KNN) e os algoritmos *conformal anomaly detection* (CAD) e *inductive conformal anomaly detection* (ICAD). O KNN-CAD realiza a detecção em duas etapas principais: (i) cria uma representação da série usando a técnica de janelas deslizantes e (ii) aplica nessa representação uma análise baseada em distância com o algoritmo KNN para mensurar se as observações analisadas estão em conformidade com um limite (fronteira) da distância esperada para os demais dados da série [Salles et al., 2020].

Com foco na detecção de eventos *online*, na literatura há diversos métodos aplicáveis. Uma visão geral sobre o estado da arte nessa área é apresentada por Ariyaluran Habeeb et al. [2019] e adaptada por Ogasawara et al. [2021]. Os próximos trabalhos apresentam métodos voltados especificamente para detecção *online*.

Rettig et al. [2015] apresentaram um método para detecção *online* de eventos visando escalabilidade e generalização e usando as ferramentas de processamento de *streaming Apache Kafka* e *Spark streaming*. O método usa as métricas entropia relativa e correlação de Pearson para avaliar os dados e identificar anomalias, sem assumir previamente uma distribuição para os dados. A

[1]Não é usado o conceito de lotes explicitamente, mas na análise de mudança de estrutura da série é feita uma adaptação do conceito de janelas deslizantes para aplicar testes estatísticas na série em *streaming*

combinação dessas métricas permite identificar: (i) mudanças graduais no *stream* de dados com a entropia relativa e (ii) mudanças abruptas com a correlação de Pearson [Rettig et al., 2015].

Ahmad et al. [2017] propuseram um método para detecção de eventos em séries temporais em *stream*. O método é baseado no algoritmo Memória Temporal Hierárquica (MTH), um *framework* teórico para aprendizado de sequências de neurônios corticais. O MTH usa aprendizado de máquina para modelar a distribuição da série e comparar novas observações com as predições do modelo, marcando eventos do tipo anomalia segundo a divergência entre os valores reais e preditos [Ahmad et al., 2017; Hasani, 2017].

Munir et al. [2019] desenvolveram o método *FuseAD* que detecta eventos do tipo anomalia em diferentes tipos de séries temporais aplicando aprendizado não supervisionado. Para tanto, o *FuseAD* combina métodos estatísticos e aprendizado de máquina, usando o modelo ARIMA [Shumway and Stoffer, 2017] e uma CNN na etapa de predição e um detector de anomalias na etapa final. A combinação do ARIMA e CNN é usada para predição dos próximos valores e as novas observações da série são marcadas como eventos em função de sua distância em relação aos valores preditos [Munir et al., 2019].

Zhang et al. [2020] apresentaram outro método que emprega uma etapa de predição e outra de detecção baseada na divergência entre observações reais do *streaming* e as predições do modelo é apresentado. O método é denominado MF-*stacked* LSTM-EWMA, devido à combinação dos algoritmos filtro de mediana (MF), *Long Short Term Memory* (LSTM) empilhada (*stacked*) e EWMA. O método MF-*stacked* LSTM-EWMA é dividido em três etapas: (i) identificar anomalias óbvias, (ii) predizer as próximas observações da série e (iii) detectar eventos.

Para realizar as três etapas do MF-*stacked* LSTM-EWMA os algoritmos combinados são usados na seguinte sequência: MF para identificar anomalias óbvias, *stacked* LSTM como preditor e EWMA para criar fronteira com limites para identificar as divergências entre os valores preditos e as observações reais. Na etapa de predição, ao invés de uma única LSTM, é usada uma abordagem em que são empilhadas camadas de LSTM para obter as predições que serão usadas como base para detecção dos eventos [Zhang et al., 2020].

Algumas ferramentas ao invés de apresentar métodos isolados propõem *frameworks* ou serviços em plataformas para detecção de eventos. Um exemplo de serviço de detecção de eventos em uma plataforma de nuvem é o Amazon *QuickSight Machine learning powered anomaly detection* (ML-AD)[2] da plataforma *Amazon Web Services* (AWS). A documentação do serviço na AWS tem foco na forma de uso e aplicações, mas não foram identificadas publicações com detalhes sobre o método usado por esse serviço para comparação neste trabalho.

Ren et al. [2019] implementaram um serviço denominado Microsoft Anomaly Detector (MAD)[3]

[2]Disponível em: https://docs.aws.amazon.com/quicksight/latest/user/anomaly-detection.html
[3]Disponível em: https://azure.microsoft.com/en-us/services/cognitive-services/anomaly-detector/

na plataforma de nuvem *Microsoft Azure*. O MAD usou o algoritmo SR-CNN, uma combinação do algoritmo *spectral residual* (SR) e uma CNN [Ren et al., 2019]. O SR realiza uma transformação da série gerando uma representação denominada mapa de saliência na qual os valores extremos são acentuados, enquanto a CNN usa essa versão transformada da série para aprender as fronteiras para determinar a ocorrência de eventos.

Talagala et al. [2020] propuseram um *framework* para detecção *online* de eventos em séries temporais multivariadas. Esse *framework* é baseado na teoria de valores extremos (TVE) para o cálculo de fronteiras do comportamento normal dos dados para identificar mudanças significativas em novas observações. A detecção de eventos proposta lida com a não estacionariedade por meio da divisão da série em janelas deslizantes e ajustando o modelo à medida que mudanças no comportamento dos dados são detectadas [Talagala et al., 2020].

III.2 Trabalhos de comparação de métodos

Além de trabalhos focados em propor métodos para detecção de eventos, buscou-se identificar estudos de comparação de métodos. Esse refinamento da busca inicial da seleção de trabalhos relacionados foi realizado para contribuir no entendimento dessa área de pesquisa e das metodologias adotadas para executar experimentos de comparação de métodos. Para isso, foram realizados procedimentos de filtros e de *snowballing* nos documentos retornados pela *string* de busca a procura, especificamente, de trabalhos que possuem títulos contendo os termos *survey*, *benchmark* e *comparison*, adicionando mais 9 documentos àqueles avaliados na seção anterior.

Esta seção apresenta esses trabalhos de comparação de métodos com a descrição de como foram realizadas as comparações, métricas e conjuntos de dados usados e métodos melhor avaliados. Apesar de muitos dos trabalhos não apresentarem *frameworks* ou ferramentas que habilitem a adaptação dos métodos ao *streaming*, representam uma linha de base nessa área de pesquisa. Assim, a identificação de aspectos positivos, lacunas ou necessidades de complementação são comparadas, ou melhoradas com a metodologia proposta neste trabalho.

Salles et al. [2020, 2023a] desenvolveram o *framework Harbinger* que disponibiliza funções para detecção de eventos, avaliação, visualização, combinação de métodos e comparação das detecções realizadas. Ao invés de um método específico, o *Harbinger* possibilita a implementação e a combinação de diferentes métodos de detecção de eventos, mas os executa de maneira *offline*. A pesquisa analisou sete métodos por meio de suas implementações no *Harbinger*, usando a métrica F_1 para comparar o desempenho das detecções, tendo apontado resultados superiores para os métodos AN, decomposição, KNN-CAD e EWMA [Salles et al., 2020].

A pesquisa de Ahmad et al. [2017] fornece um *framework* denominado Numenta, o conjunto de dados *Numenta Anomaly Benchmark* (NAB) e a métrica NAB *Score* e experimentos para compa-

ração de métodos de detecção de eventos *online*. Diversos métodos foram comparados na detecção *online* e os três métodos com informações publicadas com melhores desempenhos foram MTH, *Relative Entropy* e *Twitter ADVec* [Ahmad et al., 2017]. Os métodos foram comparados usando as métricas NAB *Score* e tempo de latência em micro segundos.

Hasani [2020] apresenta uma análise de métodos para detecção *online* de eventos e uma comparação do desempenho, tempo de execução e uso de CPU. Foram analisados os métodos ARIMA, médias móveis, HW e HW-GA [Hasani, 2020]. Para o cálculo das métricas foi realizado monitoramento em tempo real do uso de CPU e tempo de execução de cada método [Hasani, 2020].

Belacel et al. [2022a] apresentam uma prova de conceito do uso de métodos existentes nos pacotes *Scikit-Multiflow* e *River* integrados à plataforma de *streaming Kafka*. Os métodos aplicados foram *Half-Space Trees* (*HS-Trees*) e *Isolation Forest* (*IForestASD*), contudo o trabalho não explora o funcionamento dos métodos em si, mas os aspectos relacionados à sua integração com a plataforma *Kafka*. As métricas usadas para comparação dos métodos foram precisão, revocação, F_1 *score*, suporte e tempo de execução. As principais contribuições do trabalho referem-se à proposta de um modelo de integração de métodos ao Kafka e redução do tempo de execução em cenários de *streaming* quando comparado à detecção *offline*, mesmo mantendo níveis aceitáveis nas métricas de detecção.

Na análise dos trabalhos de comparação de métodos percebe-se o uso de métricas tradicionais de tarefas de classificação [Han et al., 2012; Salles et al., 2020] e custo computacional [Hasani, 2020]. Também são usadas algumas métricas adaptadas para o contexto de séries temporais [Ahmad et al., 2017; Escobar et al., 2021].

A análise da forma de seleção de conjuntos de dados, por sua vez, apresenta algumas discussões sobre cuidados na seleção de conjuntos de dados. Aborda-se também fragilidades existentes em conjuntos amplamente usados na literatura [Wu and Keogh, 2021; Lomio et al., 2020]. A identificação de conjuntos de dados considerados *benchamark* na detecção e suas fragilidades contribui para experimentos abrangentes.

Quando se trabalha com detecção de eventos *online* em séries em *streaming*, não há rótulos disponíveis, uma vez que cada observação ou conjunto de observações novas é analisada em fluxo contínuo. Contudo, para comparar métodos é normal o uso de métricas comuns em tarefas de classificação que dependem de rótulos para seu cálculo. Isso ocorre em métricas calculadas a partir da matriz de confusão como acurácia, precisão, revocação e F_1 que são obtidas pela comparação dos eventos reais (rotulados) com aqueles detectados [Han et al., 2012; Escobar et al., 2021; Ahmad et al., 2017].

Os cálculos para gerar a matriz de confusão e sua dependência de rótulos foi discutida em detalhes na Seção II.5. A inexistência de rótulos é comumente contornada na criação de métodos

com ferramentas de comparação que usam conjuntos de dados rotulados como *benchmark*. As métricas relacionadas ao desempenho computacional do método são mais simples de analisar por não dependerem de rótulos.

As métricas de desempenho computacional comuns são tempo de execução, latência, consumo de memória e CPU. A análise desse tipo de métrica compreende apenas seu registro pelo monitoramento das execuções e comparação para identificar os métodos com maior ou menor valores [Hasani, 2020; Ahmad et al., 2017]. Ainda que seja simples, a comparação, por exemplo, do tempo de execução está relacionada diretamente à capacidade de reposta tempestiva a efeitos indesejados nos processos geradores das séries.

A métrica NAB *Score*, criada por Ahmad et al. [2017], parte de cálculos das métricas de classificação, mas inclui uma forma de recompensar detecções precoces e penalizar detecções tardias. Para isso é calculada uma janela temporal ao redor do ponto no tempo onde o rótulo indica a existência de um evento. Nessa abordagem, os verdadeiros e falsos positivos não são calculados pelo acerto ou erro exato de um ponto no tempo, mas pela indicação de um evento dentro da janela calculada na métrica. Além disso, detecções logo no início da janela tem maior peso.

Uma análise de métodos com foco na avaliação de viés temporal é apresentada por Escobar et al. [2021]. O viés temporal está relacionado à identificação da tendência de métodos de antecipar ou atrasar as detecções de eventos. O trabalho discute que métricas tradicionais baseadas no acerto dos pontos exatos dos eventos são insuficientes para esse tipo de análise. Além disso, é levantada a questão que para algumas aplicações pode ser desejável que a detecção seja antecipada ao invés de ocorrer apenas em um ponto exato da série [Singh and Olinsky, 2017; Escobar et al., 2021].

Salles et al. [2023b] propõem um conjunto de métricas denominadas *SoftED*[4] que suavizam os efeitos de métricas de classificação tradicional para incorporar tolerância na imprecisão de detecção de eventos. O objetivo relacionado à tolerância temporal das métricas *SoftED* é melhorar a análise de detecções de eventos, associando eventos rotulados às detecções em momentos do tempo próximos ao ponto exato do rótulo. Para obter o efeito desejado, o cálculo dos valores de VP, VN, FP e FN são adaptados para considerar a tolerância temporal, com isso métricas baseadas nesses valores como acurácia, precisão, revocação, F_1 incorporam essa tolerância.

Lomio et al. [2020] apresentam um conjunto de dados denominado *dataset for cloud-native memory anomalies* (RARE) sobre monitoramento de uso de memória em serviços de nuvem. O objetivo da publicação do RARE não foi a avaliação de métodos, mas suportar outros pesquisadores a realizarem essa tarefa com provimento de dados. Por isso, sua comparação foi com conjuntos de dados Yahoo *Labs*[5] e NAB [Ahmad et al., 2017][6] que contém dados sintéticos e reais sobre

[4]Disponível em: https://github.com/cefet-rj-dal/softed
[5]Disponível em https://yahooresearch.tumblr.com/post/114590420346/
[6]Disponível no pacote R OTSAD

monitoramento de serviços de nuvem com rótulos de eventos.

Outro conjunto de dados normalmente usado para avaliar métodos de detecção de eventos é o Gecco *Challenge*[7] que também contém dados rotulados sobre qualidade da água [Moritz et al., 2018]. No Gecco as variáveis há eventos reais e sintéticos adicionados para uso em testes de detectores de evento. O conjunto contém 9 séries que podem ser analisadas em conjunto como série multivariada ou individualmente como séries univariadas.

Confrontando os resultados de muitos trabalhos sobre detecção de eventos, Wu and Keogh [2021] afirmam que conjuntos de dados amplamente usados nessa área como NAB e Yahoo *Labs* apresentam fragilidades graves que geram resultados não confiáveis e ilusão de progresso. As fragilidades são classificadas em: (i) trivialidade, (ii) densidade irreal de anomalias, (iii) área verdadeira (verdadeiros positivos e verdadeiros negativos) rotulada erroneamente e (iv) viés de execução até de falha. Para lidar com essas fragilidades Wu and Keogh [2021] apresentam o conjunto de dados *University of California, Riverside* (UCR) *Anomaly Archive* composto de diversas séries com eventos rotulados que afirma evitar as falhas citadas.

Ao comparar métodos, além dos aspectos já citados, é importante analisar requerimentos desejáveis para detecção de eventos em tempo real [Stahmann and Rieger, 2021]. Stahmann and Rieger [2021]; Vyas et al. [2021] indicam como exemplos de requerimentos desejáveis tanto dos métodos quanto dos dados e da forma de execução: a origem e composição adequada dos dados, capacidade de resposta, escalabilidade, baixa latência, tolerância a falhas e pós-processamento (avaliação, notificação e ajustes). São requerimentos específicos dos métodos de detecção: capacidade de aprendizado e adaptação contínua, lidar com séries não estacionárias e mudanças em geral ao longo do tempo, detecção precoce, minimização de falsos positivos e falsos negativos [Ahmad et al., 2017].

III.3 Síntese de trabalhos relacionados

Uma visão dos trabalhos relacionados, conforme análise das Seções anteriores deste capítulo, é sintetizada nesta seção para reforçar pontos relevantes observados, lacunas e aspectos relacionadas à evolução da pesquisa. A Figura III.1 ilustra a distribuição ao longo dos anos dos documentos obtidos a partir da *string* de busca mencionada no início deste capítulo. Observa-se um crescimento expressivo nos últimos anos da produção acadêmica sobre detecção *online* de eventos em séries temporais apresenta, especialmente a partir de 2011.

Esse crescimento reforça a importância da pesquisa sobre detecção *online* de eventos em séries temporais. Avaliando os últimos anos, há uma leve redução entre 2019 e 2020. Em 2022 o número de publicações volta a crescer significativamente.

É importante na comparação dos métodos a escolha adequada de uma linha de base que iden-

[7]Disponível no pacote R EventDetectR

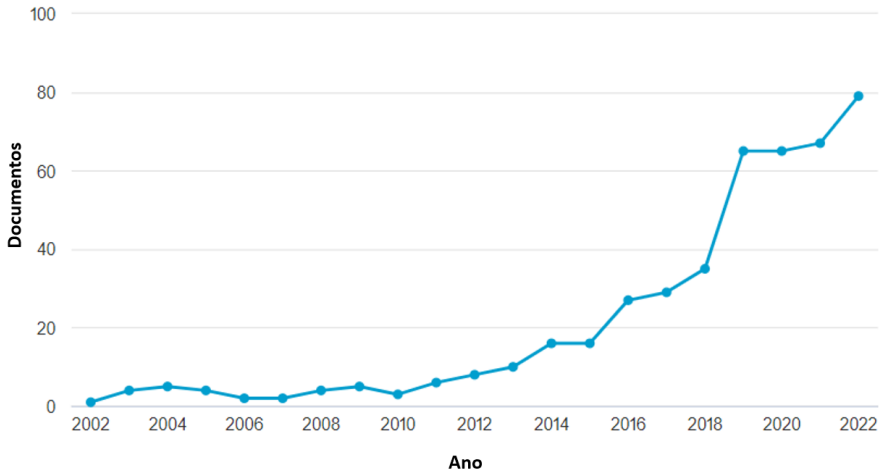


Figura III.1: Evolução da quantidade de publicações: Quantidade de trabalhos sobre detecção de *online* publicados por ano até 2023. Fonte: Análise do resultado da *string* de busca na base *Scopus*

tifique o desempenho mínimo de um método e o estado da arte a ser superado. Deve-se atentar para métodos que já funcionem adequadamente mesmo na detecção *offline* [Carmona, 2013; Hasani, 2017; Dancho and Vaughan, 2020; Guralnik and Srivastava, 1999; Takeuchi and Yamanishi, 2006; Escobar et al., 2021]. Esses aspectos são tratados em maiores detalhes na avaliação experimental.

Dessa síntese de trabalhos relacionados, também é possível intuir a necessidade de confrontação com trabalhos de comparação de métodos. Quanto a esses trabalhos que comparam métodos foram percebidos como relevantes os três aspectos a seguir: (i) forma de execução dos métodos, (ii) conjuntos de dados usados e (iii) métricas para comparação dos métodos.

Percebe-se uma concentração no uso dos conjuntos de dados NAB e Yahoo *Labs*. Essa concentração, por um lado reforça a importância dessas séries, mas pode gerar fragilidades como excesso de especialização de métodos e risco de baixa capacidade de generalização. Por isso, são importantes as discussões que levaram a proposição do RARE [Lomio et al., 2020] e UCR *Anomaly Archive* [Wu and Keogh, 2021] que complementam ou contestam, respectivamente, as séries de uso predominante para avaliação de métodos.

Algumas recomendações de Wu and Keogh [2021] são severas, como abandonar os conjuntos de dados mais usados na área de pesquisa e rever quaisquer publicações que os tenha como principal base de dados. A falha de *área verdadeira rotulada erroneamente*, por exemplo, é baseada na análise visual das regiões das séries com anomalias e na incoerência dos rótulos contidos nas séries. Essa técnica (análise visual) é importante para explorar a série, mas não garante que todas as séries refutadas por Wu and Keogh [2021] repitam os problemas dos intervalos analisados. De qualquer modo, mesmo que não se desconsidere totalmente o uso de séries como NAB e Yahoo *Labs*, fica evidente a importância da seleção adequada de múltiplos conjuntos de dados para avaliação de métodos de detecção eventos.

O uso de lotes para processamento de dados tem aplicações tanto para tarefas de processamento *offline* quanto *online*. Normalmente há uma associação direta de processamento *offline* com o termo processamento em lote, mas essa associação é mais adequada quando se refere ao acesso prévio aos dados completos [Truong et al., 2020]. Contudo, como foi visto, o processamento de dados em *streaming* pode ocorrer por meio do uso de pequenos lotes [Rettig et al., 2015; Hiraman et al., 2018].

Essa abordagem de pequenos lotes é similar ao conceito de subsequências ou janelas deslizantes [Fu, 2011; Lin et al., 2003]. A divisão de séries temporais em lotes foi experimentada para o processamento em *streaming* [Iwashita and Papa, 2019]. Contudo, Iwashita and Papa [2019] não aplicaram sua abordagem à detecção de eventos, mas à identificação de mudanças de conceito.

Além do funcionamento das detecções, alguns trabalhos abordam a integração com plataformas de nuvem e ferramentas de processamento de *streaming*, aspectos adicionais relevantes para detecção *online* de eventos. Sobre a integração com plataformas de nuvem foram citados MAD [Ren et al., 2019] e *Amazon* ML-AD e sobre ferramentas de processamento de *streaming* os trabalhos de Rettig et al. [2015] e Belacel et al. [2022a].

Uma síntese dessa análise dos trabalhos relacionados é apresentada na Tabela III.1. Nessa síntese abordam-se as características e limitações, como disponibilização de ferramentas para experimentação de métodos existentes e capacidade de adição de novos. Adicionalmente, a Tabela III.1 apresenta informações relacionadas ao processamento da série em *streaming* como dependência de alguma plataforma, uso de lotes e fornecimento de métricas ou informações agregadas para análise específica do comportamento dos métodos no *streaming*.

Para classificação na Tabela III.1 considera-se a disponibilidade de ferramenta de experimentação quando a implementação já possui nativamente formas de experimentar métodos existentes. Quando o trabalho apresenta apenas uma metodologia ou um pacote com métodos pré-definidos foi considerada indisponibilidade de ferramenta. A possibilidade de adicionar novos métodos é avaliada de maneira análoga, ou seja, é preciso que a própria ferramenta tenha mecanismo pronto para inclusão de métodos, como ocorre com Numenta e Harbinger [Ahmad et al., 2017; Salles et al., 2023a].

No geral, os trabalhos de comparação de métodos listados na Tabela III.1 não fornecem ferramentas ou *frameworks* prontos para experimentação de novos métodos ou daqueles já existentes. Há exceções, onde isso é possível, em alguns trabalhos, como Harbinger [Salles et al., 2020, 2023a], Numenta [Ahmad et al., 2017] e MAD [Ren et al., 2019]. Contudo, no caso do MAD a adição de novos métodos só é possível por meio de serviços pagos na plataforma *Azure*.

Os trabalhos que não permitem experimentação de novos métodos normalmente focam na comparação e apresentação da própria metodologia como feito por Belacel et al. [2022a] e Hasani [2020].

Tabela III.1: Trabalhos de comparação de métodos: Características e limitações. Título dos trabalhos: Na primeira coluna é informado o nome do artefato principal do trabalho ou dos autores, quando não há um artefato com um nome atribuído.

Trabalho	**Ferramenta de Experimentação**	**Adição de Novos de Métodos**	**Plataforma**	**Uso de Lotes**	**Avaliação do Proc. em *Streaming***
Numenta	Sim	Sim	Independente	Não	Latência
Harbinger	Sim	Sim	Independente	Não	Não
MAD	Sim	Apenas serviços pagos	Azure	Não	Não
Amazon ML-AD	Não	Não	AWS	Não	Não
Strucchange	Não	Não	Independente	Parcial	Não
[Rettig et al., 2015]	Não	Não	Kafka e Spark	Parcial	Não
[Hasani, 2020]	Não	Não	Independente	Não	Tempo de execução, Uso de CPU
[Belacel et al., 2022b]	Não	Não	Kafka, Scikit Multiflow e River	Não	Não

Outros como o trabalho apresentado por Rettig et al. [2015] ou o pacote R Strucchange apesar de ter essa limitação, não fazem apenas comparação, mas fornecem métodos próprios aptos para uso. Essas limitações indicam lacunas, mas os trabalhos são úteis para entendimento da integração com plataformas de nuvem ou de processamento em *streaming* que podem ser objeto de trabalhos futuros.

A dependência de plataformas identificada em alguns trabalhos pode ser considerada uma limitação. Essa interpretação justifica-se especialmente quando há dependência de serviços pagos como nas ferramentas MAD e Amazon ML-AD. Entretanto, a integração com plataformas de *streaming* como nos trabalhos de Rettig et al. [2015], Hasani [2020] e Belacel et al. [2022a] contribuem como base para trabalhos futuros.

O uso de lotes como parte do treinamento e detecção não foi identificado de maneira explícita nos trabalhos estudados. Mas, há métodos - identificados como uso parcial na Tabela III.1 - que aplicam adaptações ou conceitos similares. Como exemplos há o pacote *Strucchange* cujo uso de janelas deslizantes foi mencionado anteriormente e o método desenvolvido por Rettig et al. [2015] que aproveita o processamento de lotes da plataforma *Spark*.

Outro aspecto importante para identificar lacunas na literatura é que a maioria dos trabalhos não aborda métricas ou informações específicas para avaliação do processamento em *streaming*. Como se percebe na Tabela III.1 é comum o uso das métricas de detecção adaptadas de tarefas

de classificação. Mesmo na avaliação do *streaming*, quando isso é feito, usa-se de maneira geral informações do desempenho computacional como tempo de execução, latência e consumo de CPU.

Os trabalhos analisados neste capítulo apresentam diversidade de métodos para detecção de eventos *online*, aplicação de métricas para comparação dos métodos, além de conjuntos de dados e ferramentas para processamento de séries em *streaming*. Contudo, segundo análise dos trabalhos relacionados, ainda há carência de trabalhos focados especificamente na comparação de métodos de detecção de eventos em dados em *streaming*.

Não se observa exploração abrangente das possibilidades do uso lotes deslizantes de diferentes, exceto pela similaridade do conceito usado em janelas deslizantes. Também identifica-se deficiência quanto às métricas de análise do processamento na detecção em *streaming*. Este trabalho explora essas questões por meio da metologia apresentada no próximo capítulo.

Capítulo IV Metodologia

Este capítulo apresenta a metodologia para análise dos impactos do uso de lotes deslizantes na detecção de eventos em séries temporais em *streaming*. A metodologia é concretizada na apresentação de um *framework* denominado *Nexus*. A metodologia é exposta para detalhar como são integrados e avaliados métodos de detecções de eventos em série em *streaming* por meio do *Nexus*.

Para melhor entendimento dos detalhes da metodologia, este capítulo é dividido em quatro seções. A Seção IV.1 apresenta a forma para análise de probabilidade e de atraso nas detecções em *streaming*. A Seção IV.2 aborda a lógica geral do *Nexus* e seu diagrama de classes. A Seção IV.3 discute as abordagens para tratamento da memória dos dados e particularidades do uso de lotes deslizantes para detectar eventos. O capítulo se encerra com a Seção IV.4 que detalha a implementação do *Nexus* usando a linguagem R.

IV.1 Probabilidade e atraso na detecção de eventos em séries temporais em *streaming*

Esta seção discute a análise de probabilidade e de atraso das detecções em *streaming*. Essa discussão trata a lacuna identificada quanto à disponibilidade de formas de mensurar o comportamento dos métodos ao longo do *streaming* de séries temporais. Apesar de inspirados nos conceitos tradicionais de análise de probabilidade [Han et al., 2012; Shumway and Stoffer, 2017] e na análise de atraso nas detecções [Talagala et al., 2020], as métricas apresentadas são inovações propostas por este trabalho.

Ao longo deste trabalho o termo evento é usado para identificar observações em uma série que não condizem com o comportamento normal dos dados. Para analisar a probabilidade relacionada ao grau de assertividade com que uma detecção foi realizada ao longo do *streaming*, nesta seção usar-se-á o termo evento amostral simples quando o texto a seguir se referir a um evento estatístico [Han et al., 2012; Shumway and Stoffer, 2017]. Essa diferença de termos se faz necessária para evitar ambiguidade no entendimento da formalização apresentada a seguir.

Dada uma série temporal X, a probabilidade de detecção, do inglês *detection probability* (DP) é diretamente proporcional à quantidade de vezes em que um ponto da série é marcado como evento pelo *Nexus* nos lotes deslizantes. Para a avaliar DP é preciso mapear cada observação x_i

presente nos lotes durante o processamento. Assumindo que x_i aparece em lotes b_j, onde $j = \frac{i}{s}$, x_i é disponível para detecção $bf(x_i)$ vezes. A Equação IV.1 caracteriza a frequência bf para uma observação x_i, para memória parcial ou completa.

$$bf(x_i) = \begin{cases} m, & \text{para memória parcial} \\ \lceil \frac{n}{s} \rceil, & \text{para memória completa.} \end{cases} \tag{IV.1}$$

Seja B_i o conjunto em que a série X é dividida nos lotes com observações x_i, então $B_i = \{b_j, ..., bj + bf(x_i)\}$. Um detector tem $bf(x_i)$ avaliações para detectar se x_i é um evento o não. Deste modo, a frequência de detecção $df(x_i)$ pode ser definida pelo número de vezes em que uma observação x_i é identificado como evento por um método M. Assim, pode-se definir a probabilidade de detecções pela Equação IV.2.

$$DP(x_i) = \frac{df(x_i)}{bf(x_i)} \tag{IV.2}$$

Para avaliar especificamente a capacidade de detecção precoce [Talagala et al., 2020] dos métodos executados com o uso do *Nexus*, é avaliado o *lag* de cada detecção realizada. Para tanto, o *lag* é considerado como o atraso na detecção, ou seja, a quantidade de lotes decorridos entre a primeira leitura do ponto e o primeiro lote em que esse ponto é detectado como evento. Formalmente, considerando um ponto x_i da série X, sb_i o início da leitura desse ponto em lotes (*start batch*) e fdb_i o primeiro lote de tamanho s em que o ponto x_i é detectado como evento (*first detection batch*), temos Lag_i^s obtido pela Equação IV.3.

$$Lag_i^s = fdb_i - sb_i \tag{IV.3}$$

Para exemplificar o uso e interpretação do Lag_i^s, pode-se considerar dois pontos hipotéticos nos momentos de tempo 15 e 150 da série X, analisados por um detector em lotes de tamanho 27. Se o ponto no tempo x_{15} é detectado como evento a partir do lote 3, dado que esse ponto está disponível desde o lote 1, temos $Lag_{15} = fdb_{15} - sb_{15}$, ou seja, $Lag_{15} = 3 - 1 = 2$, portanto, para o ponto no tempo x_{15} há atraso de 2 lotes na sua detecção como evento. O ponto $t = 150$, por sua vez, tem $sb_{150} = 6$, pois é lido pela primeira vez no lote 6. Se considerar que o ponto x_{150} é detectado como evento no mesmo lote em que é lido pela primeira vez, temos $fdb_{150} = 6$, portanto $Lag_{150} = 6 - 6 = 0$, o que indica que não há atraso na sua detecção como evento.

Esta seção apresentou a contribuição deste trabalho para interpretação do funcionamento de métodos ao longo do processo de detecção em *streaming*. Para isso é proposta a análise de eventos consolidados por meio da probabilidade $DP(x_i)$ e a existência de atrasos nas detecções ao longo dos lotes por meio do Lag_i^s. A próxima seção apresenta o *framework Nexus*, detalhando sua forma

de funcionamento para habilitar aplicação de métodos para detecção de eventos em *streaming*.

IV.2 *Nexus*: lotes deslizantes na detecção de eventos em séries temporais em *streaming*

Esta seção detalha a proposta de análise do impacto de uso de lotes deslizantes para detecção de eventos em séries temporais em *streaming*. São abordadas questões gerais da lógica de funcionamento do *framework Nexus* por meio do qual a série é percorrida em fluxo contínuo, analisando os pontos da série à medida que são lidos para classificá-los como eventos. Também se detalham as etapas do *workflow*, seu significado e possíveis adaptações da forma de funcionamento do *Nexus*.

O *Nexus* contribui para a capacidade de experimentação de métodos de detecção de eventos em dados em *streaming* por meio do uso de lotes deslizantes. Adicionalmente, contribui para (i) identificação do desempenho de diferentes métodos quanto à capacidade de identificar eventos reais e (ii) celeridade na capacidade de resposta aos eventos pela análise do tempo de execução e custo computacional. Além disso, a abordagem proposta é agnóstica a métodos ou plataformas de *streaming* específica.

O *workflow* de execução dos métodos, apresentado na Figura IV.1, recebe uma série temporal X e parâmetros adicionais Pr para execução da detecção de eventos em *streaming* na série. esses parâmetros adicionais de configuração são: o **método aplicado** (M) para detecção com seus respectivos hiper-parâmetros; o tamanho w do **subconjunto para aquecimento (*warmup*)**, ou seja, quantidade mínima de observações para início das detecções; o **tamanho do lote** s em cada iteração da execução do *streaming*; e a quantidade de lotes m que o *workflow* deve manter na **memória dos lotes**, ou seja, como deve percorrer a série e gerar a versão atualizada (W'): (i) $m \geq 1$: manter m lotes antigos; (ii): $m = 0$: manter a memória completa de todos os lotes ao longo das iterações.

A série X é processada em *streaming* por meio de lotes para treinamento dos modelos e submissão das observações do lote corrente à detecção de eventos com o método M. O processo começa pela separação de uma quantidade w de observações iniciais, denominada *warmup*, para permitir acesso à quantidade mínima de dados para iniciar as detecções. Para executar o processamento da série em *streaming*, o *workflow* prossegue, deslizando ao longo da série em lotes de tamanho s de maneira iterativa até que toda a série seja percorrida. Uma intuição desse processo de deslizar ao longo da série através dos lotes é ilustrado mais adiante nas Figuras IV.3 e IV.4 com ou sem manutenção da memória dos dados de lotes antigos, respectivamente.

Para realizar a detecção *online* de eventos, o lote mais recente sempre é o objeto das detecções. Como saída final é obtido um vetor EC com os eventos consolidados. A análise da saída EC permite avaliar o resultado das detecções à medida que os lotes são submetidos à análise ao longo do tempo.

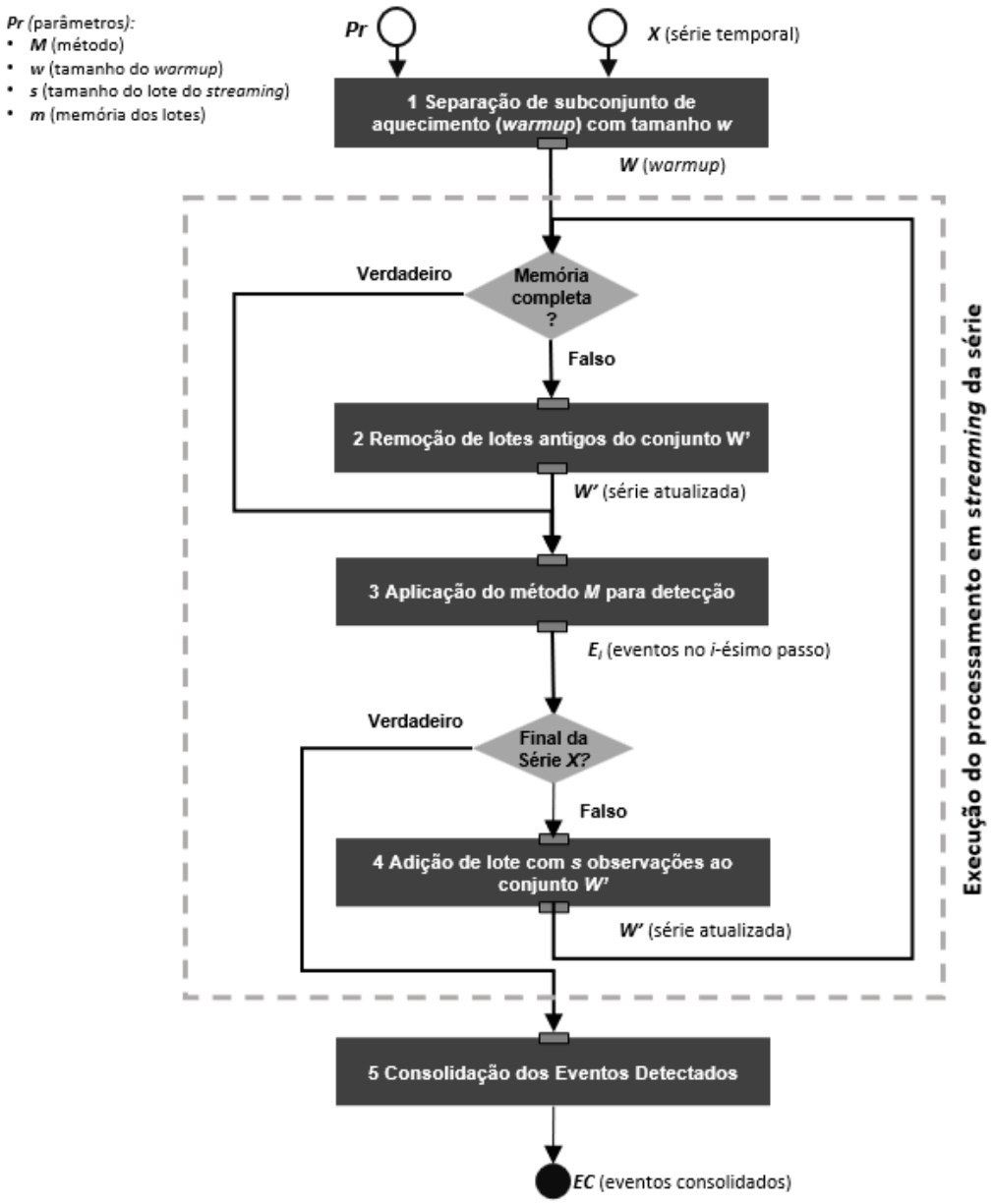


Figura IV.1: Diagrama do *Nexus* - Aplicação de métodos de detecção de eventos em séries em *streaming*

Os passos do $workflow$ proposto, cuja visão geral da lógica está na Figura IV.1, podem ser agrupados em três principais grupos de atividades como estão detalhados a seguir: (i) **separação do *warmup***; (ii) **execução do *streaming***; (iii) **consolidação dos eventos detectados**.

Primeiro, tem-se a **separação do *warmup*** (i), realizada no passo 1 da Figura IV.1. O subconjunto de *warmup*, identificado como W de tamanho w, é uma subsequência da série X, composta da observação x_1 até x_w, onde x_1 é a observação mais antiga da série X e x_w é a observação na w-ésima posição da série. O valor inteiro w determina a quantidade de observações para início das detecções.

Depois, tem-se a **execução do *streaming*** (ii), correspondente aos passos 2 a 4 da Figura IV.1,

que lê os dados em uma série em *streaming* e executa a detecção de eventos nessa série à medida que novas observações são recebidas. Para isso, inicia-se pela detecção de eventos por meio do passo 3 e a partir da segunda rodada da iteração o passo 4 adiciona s observações ao conjunto W. Essa tarefa tem como saída um novo subconjunto W' da série original X que é objeto das detecções por meio do passo 3 de maneira iterativa.

O passo 3 na primeira rodada da iteração que executa o *streaming* aplica o método para detecção no subconjunto de *warmup*. esse passo consiste em aplicar o método M, com os hiperparâmetros fornecidos, para detecção de eventos no subconjunto W (*warmup*). Como resultado desse passo é gerada a primeira lista de eventos E_i detectados no *warmup*, portanto, com $i = 1$. A avaliação dos resultados E_1 permite testar diferentes tamanhos de w.

Antes da detecção de eventos no *streaming*, se a memória estiver definida como completa, ou seja, parâmetro $m = 0$, o passo condicional 2 não é executado, o que mantém a memória dos lotes antigos. Contudo, se o parâmetro estiver definido como $m \geq 1$, o passo 2 é executado, removendo lotes antigos do subconjunto W', até que a quantidade de lotes na memória seja equivalente ao valor de m. esse passo funciona como uma forma de manter ou remover a memória de observações antigas da série no *streaming*.

Por fim, é realizada a **consolidação dos eventos detectados** (iii), no passo 5 da Figura IV.1 em que os eventos E_i são consolidados em EC. Os eventos consolidados EC contém o resultado das detecções de cada passo do *streaming*, E_i, para identificar tanto os eventos quanto o momento em que foi realizada a detecção. Os eventos consolidados são insumos para o cálculo da probabilidade de serem eventos reais da série analisada e da existência de *lag* nas detecções.

A fim de apresentar uma abstração sobre os objetos que compõem o *Nexus*, a arquitetura de classes é detalhada na Figura IV.2. Na arquitetura, a classe *StreamDetectorManager* é o núcleo do *Nexus*, responsável pela gestão da detecção em *streaming*. Essa classe tem seus próprios atributos - os parâmetros do *Nexus* - e o método que implementa a detecção em *streaming*, contudo ela é uma composição que contém as classes *Detector* e *TimeSeries*. As demais classes representam as saídas vistas no processo ilustrado na Figura IV.1 detalhada anteriormente, bem como as especializações das super classes.

Na arquitetura da Figura IV.2 a classe *Detector* implementa os métodos responsáveis por treinar modelos e executar detecções com os modelos treinados. A classe *StreamDetectorManager*, por sua vez, usa objetos das classes *Detector* e *TimeSeries* para executar a detecção de eventos em *streaming*. A relação da classe *StreamDetectorManager* é de 1 para muitos (1 *) com a classe *Detector*, tendo em vista que pode haver diversos detectores se relacionando com um único objeto da classe *StreamDetectorManager*, mas o inverso não ocorre.

A classe base *Detector* dá origem às subclasses *AnomalyDetector* e *ChangePointDetector*. As

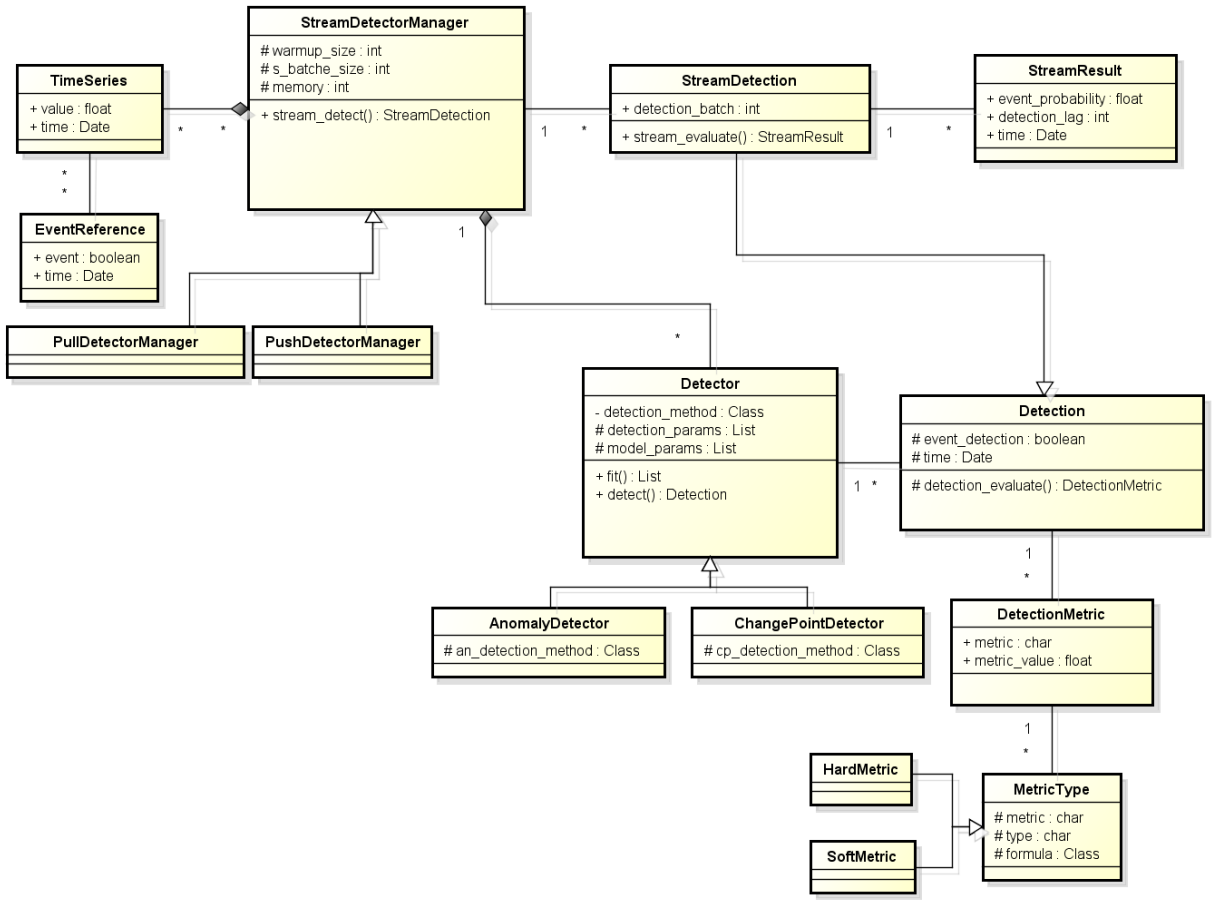


Figura IV.2: Diagrama de Classes do Nexus

subclasses são especializações que permitem aplicar métodos para detectar eventos dos tipos anomalia e pontos de mudança, previstos na taxonomia de detecção de eventos *online* [Ogasawara et al., 2021; Ariyaluran Habeeb et al., 2019]. Os atributos dos detectores são uma função *detection_ method* com o método de detecção, uma lista *detection_ params* com os parâmetros necessários para o método de detecção e uma lista *model_ params* com os parâmetros gerados a partir do treinamento do método.

As subclasses herdam os métodos *fit* e *detect* da classe base *Detector*. O método *fit* realiza o treinamento do modelo, enquanto *detect* aplica o modelo treinado para detectar eventos na série temporal identificada no diagrama da Figura IV.2 como a classe *TimeSeries*. A especialização das subclasses se dá pelos atributos *an_ detection_ method* e *cp_ detection_ method* que recebem funções de detecção de anomalias e detecção de pontos de mudança, respectivamente.

A classe *StreamDetectorManager* ela se relaciona com a entrada dos dados por meio da classe *TimeSeries* com a série temporal, que por sua vez é associada a *EventReference* que representa os rótulos dos eventos reais. Os parâmetros do *Nexus* equivalem aos atributos *warmup_ size*, o tamanho do subconjunto para *warmup*; *s_ batch_ size*, a quantidade de observações em cada lote; e *memory*, a quantidade de lotes que o executor do *streaming* manterá na memória para execução

das detecções.

Os dados da série temporal original são representados em *TimeSeries* pelos atributos *value* e *time* que registram cada valor e o momento no tempo na série, respectivamente. Os eventos reais rotulados, por sua vez, são representados em *EventReference* pelos atributos *event* e *time*. Para cada momento no tempo da série (*time*) há um registro equivalente em *time* de *EventReference*, com o valor verdadeiro no atributo *event* caso o ponto seja um evento real ou falso, caso não o seja.

A classe *StreamDetectorManager* usa os detectores criados como objetos das classes *AnomalyDetector* ou *ChangePointDetector* para aplicar a detecção de eventos em *streaming*. Para tanto, é feita a leitura da série temporal em fluxo contínuo como o método *stream_ detect*, segundo a lógica do fluxo do processo do *Nexus* apresentado na Figura IV.1, no início deste capítulo. O método *stream_ detect* gerencia a manutenção da memória dos lotes para manter sempre as configurações definidas de acesso às observações da série para treinamento e detecções.

Além disso, o método *stream_ detect* executa o treinamento do modelo por meio da chamada ao método *fit* do objeto criado a partir da classe *Detector* com base nos dados lidos do *warmup* e dos últimos lotes. Também por meio de chamada interna no método *stream_ detect*, que progressivamente é executado o método *detect* ao longo das iterações para detecção de eventos. Os resultados das detecções são alocados em um objeto da classe *StreamDetection* com as detecções de todos os lotes consolidadas.

As subclasses *PullDetectorManager* e *PushDetectorManager* são especializações que se diferenciam pela forma de iteração por meio da qual os dados são lidos à medida que os lotes são percorridos. Uma vez que as subclasses de *StreamDetectorManager* diferenciam-se apenas pela lógica de iteração, todos os atributos e métodos são herdados. A diferença de comportamento no modelo de iterador puxado ou empurrado é implementada no desenvolvimento do método *stream_ detect*.

Na classe *PullDetectorManager* aplica-se um iterador puxado, requisitando os dados de cada novo lote da série temporal até alcançar um valor nulo como resultado da requisição, indicando que não há mais observações na série para análise. Na abordagem da classe *PushDetectorManager* aplica-se um iterador empurrado, ou seja, de maneira inversa os dados são empurrados ao longo das iterações para serem processados ao invés de requisitados.

Para cálculo da probabilidade de eventos $DP(x_i)$ e Lag_i^s é aplicado o método *stream_ evaluate* da classe *StreamDetection*. Além disso, dado que *StreamDetection* é subclasse de *Detection*, herda o método *detection_ evaluate*. As detecções são comparadas com os rótulos contido em *EventReference* por meio do método *detection_ evaluate* para gerar as métricas de detecção, representadas pela classe *DetectionMetric*.

A diferenciação entre a classe base *Detection* e a subclasse *StreamDetection* é a expansão para registro de detecções por lote. Os atributos *event_ detection* e *time* registram os pontos detecta-

dos como eventos e o momento no tempo desses pontos na série, respectivamente. Para registro das detecções para cada lote é usado o atributo *detection_ batch* que existe apenas na subclasse *StreamDetection*.

Com a informação de lotes onde cada detecção ocorre, por meio da classe *StreamDetection* é possível tanto avaliar o comportamento dos métodos ao longo do *streaming* quanto obter informações agregadas específicas para detecções em *streaming* como $DP(x_i)$ e Lag_i^s. No *streaming* as métricas de detecção *DetectionMetric* podem ser geradas por lote para conferir o comportamento progressivo dos métodos de detecção. Quanto às informações agregadas de análise do *streaming*, $DP(x_i)$ e Lag_i^s, que são contribuições deste trabalho, são registradas na classe *StreamResult* para cada evento detectado.

Em *StreamResult* tem-se $DP(x_i)$ registrada em *event_ probability* segundo o cálculo da Equação IV.2. Lag_i^s é calculado conforme a Equação IV.3 e registrado no atributo *detection_ lag*. A informação sobre *lag* permite verificar se houve atraso em determinada detecção, contribuindo para identificação de métodos com capacidade de detecção precoce.

Uma vez que as detecções do *streaming* contém os eventos, momento no tempo da série e lote em que foram detectados, é possível usar o método *detection_ evaluate* herdado da classe base *Detection* para gerar métricas para cada lote. Para isso, cada momento no tempo da série é comparado com momento no tempo dos eventos em *EventReference*. Os resultados são representados por *DetectionMetric* com a identificação e valor das métricas de detecção.

A classe *DetectionMetric* relaciona-se com a classe *MetricType* que define a identificação, tipo e fórmula de cada métrica. As subclasses *HardMetric* e *SoftMetric* são especializações dos tipos de métricas. *HardMetric* representa métricas comuns de detecção de eventos e tarefas de classificação [Han et al., 2012], conforme explicadas Seção II.5 do Capítulo II. *SoftMetric* representa um conjunto de métricas que suavizam[1] os efeitos de métricas de classificação tradicional para incorporar tolerância na imprecisão de detecção de eventos [Salles et al., 2023b].

Esta seção apresentou a lógica geral do *Nexus* (IV.1) e sua estrutura de classes. Um aspecto central do *Nexus* é como é tratada a memória dos dados à medida que a série é percorrida continuamente por meio de lotes deslizantes. A próxima seção aprofunda essa questão da memória de dados por meio do detalhamento das abordagens previstas no *Nexus*.

IV.3 *Nexus*: abordagens para manutenção de memória dos dados no *streaming*

Para avaliar como a manutenção da memória de dados antigos influencia a acurácia dos métodos de detecção e o custo computacional da análise da série em *streaming* o *Nexus* prevê diferentes configurações da forma de leitura dos lotes. Essas configurações podem ser resumidas nas abordagens

[1]Disponível em: https://github.com/cefet-rj-dal/softed

de memória completa ou memória parcial dos lotes. Esta seção descreve ambas abordagens, a intuição por trás delas e aspectos adicionais que podem ser aprofundados na avaliação experimental.

A diferença entre a manutenção da memória dos dados completos da série para análise ou apenas dos últimos lotes é ilustrada nas Figuras IV.3 e IV.4. Em ambos casos, os lotes são adicionados de maneira contínua para que os métodos usem tanto os dados do conjunto de *warmup* quanto dos lotes em memória com base para aprendizado do comportamento esperado da série. Com acesso aos dados, os métodos são aplicados para detectar eventos, porém na abordagem de memória completa sempre são usados todos os lotes acumulados como base do aprendizado, enquanto na memória parcial o aprendizado ocorre apenas com base nos lotes mais recentes.

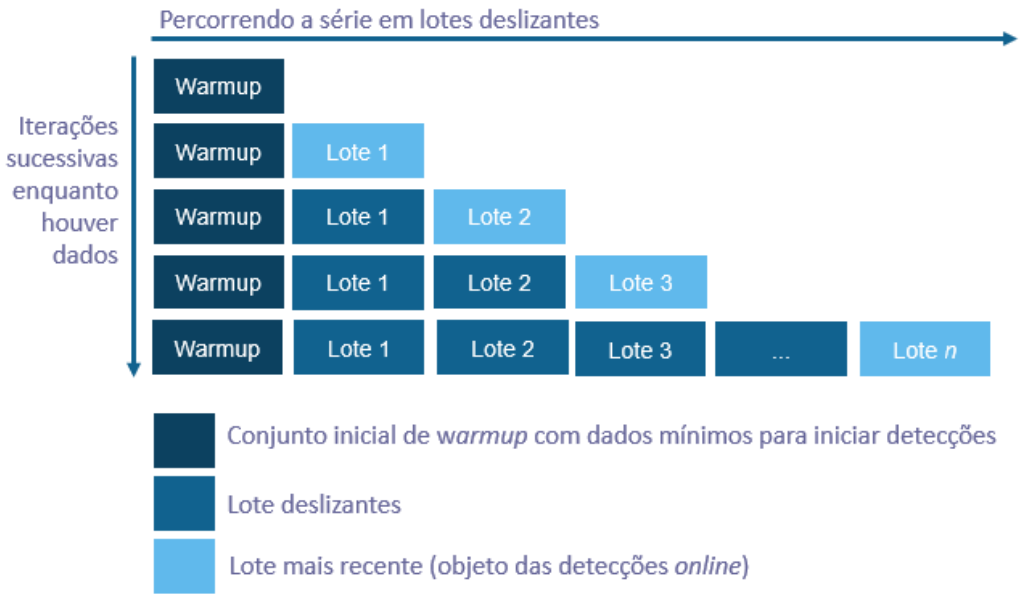


Figura IV.3: Ilustração da detecção em lotes deslizantes - Memória Completa - Adição de um novo lote a cada iteração, executando o *streaming* com memória completa dos dados

Na execução do *Nexus* mantendo a memória completa, detalhada na Figura IV.3, os lotes são adicionados continuamente e a série é analisada de maneira cumulativa. Dessa forma, há mais informações para uso pelos métodos que pode ajudar a aumentar a acurácia. Contudo, é natural ocorrer um aumento gradativo do custo computacional e eventuais falhas nas detecções pela influência de observações antigas nas detecções.

Além do custo computacional que pode aumentar muito ou até gerar situações impeditivas em séries muito longas, não há garantia que sempre haverá uma acurácia maior pelo uso da série completa como base para treinamento dos métodos. O acesso completo a dados muito antigos da série, por exemplo, pode dificultar a identificação de anomalias ocorrendo em tempo real relacionadas que podem ser relevantes dependendo da área de domínio. Portanto, a comparação dos melhores resultados em ambas abordagens depende de uma avaliação experimental ampla que use diferentes conjuntos de dados.

Quando o *Nexus* é executado com memória parcial, os lotes antigos são removidos à medida que novos lotes são adicionados (Figura IV.4). Assim, permanecem na memória apenas os lotes mais recentes equivalentes ao tamanho da memória m definida nos parâmetros de execução. Espera-se avaliar nos experimentos se essa opção gera o efeito inverso à preservação da memória, ou seja, menor custo computacional, maior influência das observações recentes na detecção de eventos e seus efeitos na acurácia da detecção.

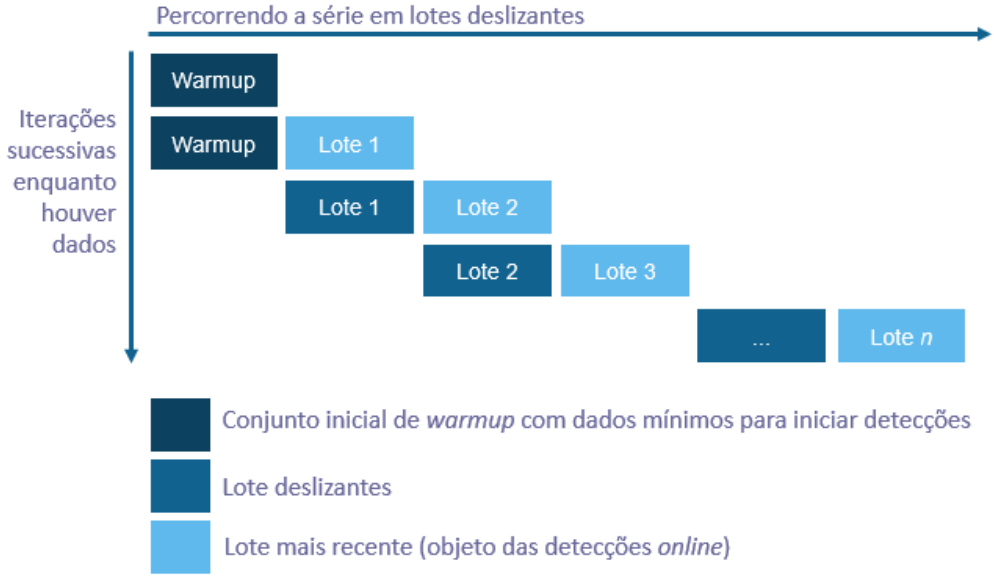


Figura IV.4: Ilustração da detecção em lotes deslizantes - Memória Parcial - Adição de um novo lote a cada iteração, executando o *streaming* removendo os lotes mais antigos

Custos computacionais menores influenciam fatores relevantes como o próprio custo financeiro, velocidade de execução dos métodos e, consequentemente, tempo de reação a efeitos indesejados ocorrendo na série temporal. Contudo, se esses benefícios comprometerem a capacidade de detecção acurada de eventos reais, os resultados podem ser inúteis para a área de domínio que precisa analisar a série temporal. A intuição por trás da memória parcial é permitir que sempre haja dados para embasar o aprendizado dos métodos, mas usar a remoção de dados mais antigos para manter o equilíbrio entre custo computacional e a acurácia dos métodos.

Na Figura IV.4 é possível perceber a intuição da memória parcial de maneira detalhada, através do exemplo de uma execução com memória de 2 lotes ($m = 2$). No sentido horizontal está ilustrado o processo de dividir a série em lotes, executar sua leitura e aplicar a detecção percorrendo-a completamente da direita para esquerda. O sentido vertical ilustra as etapas de iteração do *workflow* da Figura IV.1, usado inicialmente o *warmup* e o lote 1, depois os lotes 1 e 2 e assim sucessivamente até a leitura completa da série.

A intuição por trás da Figura IV.3 de como a série é percorrida por lotes em sentido horizontal e como esses são acumulados nas iterações do *workflow* em sentido vertical segue lógica similar

àquela apresentada na abordagem de memória parcial. A diferença fica evidente pela comparação das últimas linhas na horizontal ao final das iterações sucessiva, em uma abordagem vê-se todos os lotes mantidos na memória, enquanto na outra apenas a quantidade definida na memória parcial.

Com esta seção concluem-se os aspectos teóricos, a lógica de funcionamento e questões relacionadas à avaliação e interpretação dos resultados de detecções por intermédio do *Nexus*. A seção seguinte trata dos aspectos relativos à sua implementação para uso prático. Dessa forma, a junção das seções deste capítulo permitem tanto o entendimento das contribuições para a pesquisa na área de detecção de eventos em *streaming* quanto à reprodutibilidade e aproveitamento dos artefatos gerados.

IV.4 *Nexus*: Implementação

Esta seção tem foco nos aspectos práticos da implementação do *Nexus*. Seu objetivo é esclarecer a forma de implementação da proposta dessa metodologia e facilitar sua reprodutibilidade por outros pesquisadores. Assim, a seguir tem-se a explicação de como a arquitetura de classes e conceitos das demais seções do capítulo foram implementados na linguagem R.

O *Nexus* foi elaborado de modo a permitir experimentação e comparação de métodos distintos de maneira flexível. A implementação tem como base a arquitetura de classes da Figura IV.2, detalhada na seção anterior. Sua reprodução é possível por meio do código disponível na plataforma GitHub[2].

A classe *Detector* é implementada por meio de uma evolução do *framework Harbinger*[3] [Salles et al., 2020, 2023a]. As subclasses são especializações que permitem aplicar métodos para detectar eventos dos tipos anomalia e pontos de mudança, previstos na taxonomia de detecção de eventos *online* [Ogasawara et al., 2021; Ariyaluran Habeeb et al., 2019].

Os resultados das detecções são registrados com as detecções consolidadas, cálculo da probabilidade de eventos $DP(x_i)$ e variáveis para cálculo do Lag_i^s. Adicionalmente, na implementação há funções auxiliares para avaliar os resultados, tanto por meio do cálculo de métricas de detecção quanto da análise visual da série original e respectivas detecções. No repositório GitHub, além do acesso à implementação em R há exemplos de uso[4].

No Capítulo V são apresentados experimentos e avaliados seus resultados. Os experimentos exploraram as diferentes classes de detectores e os tipos de métodos previstos nas subclasses. Além disso, são exploradas diversas configurações dos parâmetros do *Nexus* para comparação dos métodos avaliados.

[2]Disponível em: https://github.com/cefet-rj-dal-dev/harbingerext/blob/main/develop/nexus.R
[3]Disponível em: https://github.com/cefet-rj-dal/harbinger
[4]Exemplo de uso do *Nexus*: https://github.com/cefet-rj-dal-dev/harbingerext/blob/main/develop/nexus_example.ipynb

Capítulo V Avaliação Experimental

Este capítulo trata dos experimentos realizados para avaliar a proposta de uso de lotes deslizantes na detecção *online* de eventos em séries em *streaming*. Na Seção V.1 são apresentadas as configurações usadas para realizar os experimentos. Como exemplo ilustrativo, a Seção V.2 apresenta a análise exploratória da série *pH* do conjunto de dados *Gecco Challenge*, com a identificação da presença de eventos e ilustração do comportamento de diferentes métodos e configurações de parâmetros do *Nexus*. A Seção V.3 aprofunda as possibilidades de uso do *Nexus* em séries com comportamento e tipos de evento distintos, representação de várias áreas de domínio e diferentes complexidades da tarefa de detecção.

V.1 Configurações dos experimentos

Para realização dos experimentos o *Nexus* foi implementado usando a linguagem R[1], conforme detalhes de implementação tratados na Seção IV.4. Os experimentos foram executados em um computador com CPU Intel i7-7820X e 128 GB de memória principal. Os conjuntos de dados foram carregados em memória a partir de um repositório na plataforma GitHub[2].

Os métodos de detecção selecionados são CF, FBIAD, ARIMA, GARCH e LSTM. Foram selecionados métodos que empregam desde técnicas estatísticas clássicas até aprendizado de máquina. Essa escolha tem objetivo de explorar uma diversidade adequada de abordagens de detecção de eventos. Quanto aos tipos de eventos, o método CF detecta pontos de mudança, GARCH detecta anomalias de volatilidade, enquanto os demais métodos detectam anomalias em geral.

Os métodos selecionados foram criados inicialmente para detecção *offline* de eventos. Isso permite a avaliação do *Nexus* na viabilização do uso de métodos *offline* em cenários de detecção *online*. Uma vez que métodos mais simples tendem a um custo computacional menor, essa análise proporciona uma contribuição relevante para detecção de eventos *online*.

Os parâmetros usados para cada método são exibidos na Tabela V.1. esses parâmetros representam o padrão implementado pelo *framework Harbinger* [Salles et al., 2023a] para cada método. Nos experimentos não foi realizada exploração de valores diferentes de parâmetros dos métodos para manter uniforme seu comportamento e variar apenas os parâmetros relacionados à sua execução *on-*

[1]Código disponível em: https://github.com/cefet-rj-dal-dev/harbingerext/blob/main/develop/nexus.R
[2]Disponível em: https://github.com/cefet-rj-dal/event_datasets

line por meio do *Nexus*. Desse modo, os experimentos permitem avaliar de maneira mais adequada o impacto do uso de lotes deslizantes nos resultados das detecções com cada método, com tamanhos de lotes diferentes e com preservação da memória completa ou parcial dos dados dos lotes ao longo da execução.

Tabela V.1: Parâmetros por Método

Método	**Parâmetro**	**Valor**	**Tipo**
FBIAD	*sw* (tamanho da janela deslizante)	30	Anomalia
ARIMA	*mdl* (modelo)	auto arima	Anomalia
	mdl $ARIMA(p, d, q)$	$ordem(1, 6, 2)$	
CF	*mdl* (modelo)	Regressão linear	Ponto de mudança
	ma (tamanho da média móvel)	30	
GARCH	*variance.model*	model = sGARCH	Anomalia de volatilidade
	variance.model	garchOrder = $(1, 1)$	
	mean.model	armaOrder = $(1, 1)$	
	mean.model	Incluir média = Verdadeiro	
	distribution.model	Distribuição normal	
	alpha	$1,5$	
LSTM	*inputsize*	4 neurônios	Anomalia
	epochs	10000 épocas	
	normalização	Time Series Global Min-Max	

Uma vez que os métodos empregados tem diferentes formas de funcionamento, muitos parâmetros não tem equivalência ou similaridade. Contudo, quando mais de um método usa parâmetros similares, os valores são mantidos iguais para melhor comparabilidade dos resultados. Na Tabela V.1 é possível verificar parâmetros similares com valores iguais nas linhas de *sw* e *ma* para os métodos FBIAD e CF, respectivamente, bem como os parâmetros *alpha* para os métodos FBIAD e GARCH.

Os parâmetros usados pelos *Nexus* são tamanho do *warmup*, tamanho do lote e memória de lotes (w, s, m). Os valores usados nos experimentos estão listados na Tabela V.2. Para os valores de *warmup* e tamanho dos lotes, dados em número de observações da série, as opções de valores seguem uma proporção uniforme. Para a memória de lotes, expressa em quantidade de lotes, as opções usadas são memória completa de todos os lotes e memória dos últimos três ou nove lotes.

Tabela V.2: Parâmetros do Nexus

Parâmetro	**Descrição**	**Valor**
w	Tamanho do *warmup*	N.º de observações (1,3,9,27,81,243)
s	Tamanho do lote	N.º de observações (1,3,9,27,81,243)
m	Memória de lotes	Mem. completa ou N.º de lotes (3,9)

Os valores selecionados para *warmup* e tamanho dos lotes permitem desde a execução de análise ponto a ponto (tamanho = 1) até lotes maiores com 243 observações. A gama de opções de valores foi definida para permitir a execução adequada de métodos que exigem uma quantidade mínima de observações para começar, mas também verificar se há outros que consigam começar a detectar eventos mesmo com uma quantidade pequena de observações da série. Com diferentes opções de valores também se buscou permitir experimentos com séries maiores nas quais lotes muito pequenos poderiam tornar os experimentos muito lentos e com elevado custo computacional.

Os valores para definição de tamanho *warmup* (w) e tamanho de lotes (s) iniciam em um e a partir da segunda opção seguem uma progressão uniforme de múltiplos de três. Contudo, em função de restrições da série ou dos métodos, pode não ser possível usar todas as opções de valores. Por exemplo, valores abaixo de $w = 81$, podem ser insuficientes para métodos que precisam de mais observações para o treinamento do modelo. Para equidade de condições de execução e comparação dos resultados, quando isso ocorre para determinado método, os parâmetros dos experimentos são configurados para iniciar a partir do valor mínimo que funcione em todos os métodos.

Em relação aos valores para memória de lotes, além das possibilidades de exploração da série completa pela preservação da memória de todos os lotes, foi preciso atentar para situações de incompatibilidade da memória com a quantidade de lotes no experimento. Por exemplo, a quantidade de lotes preservados precisa ser igual ou menor que a quantidade total de lotes analisadas. Também não faz sentido a quantidade de lotes ser muito próxima ao total de lotes da série, pois o efeito tanto de volume de dados para análise quanto de custo computacional seria muito próximo ou igual à execução com memória completa.

Para exploração adequada dos métodos estudados foram selecionados conjuntos de dados que representam uma ampla diversidade de áreas de domínio e comportamento das séries. A Tabela V.3 apresenta uma síntese dos conjuntos de dados usados nos experimentos. Os conjuntos de dados usados nos experimentos foram disponibilizados em um repositório público[3] para facilitar a reprodutibilidade, denominado *DAL Events Datasets*.

Tabela V.3: Conjuntos de dados

Conjunto de dados	**Quantidade de séries**	**Tipo de dados**
Gecco Challenge	9	Qualidade da água
Yahoo Labs	367	Tráfego de rede, internet e dados sintéticos
NAB	58	Serviços de nuvens e dados sintéticos
RARE	7062	Consumo de memória em serviços de nuvem
UCR Archive	250	Dados sintéticos
UCI 3W	649	Perfuração de poços de petróleo

[3]Disponível em: https://github.com/cefet-rj-dal/event_datasets

O repositório público DAL *Events Datasets* contém tanto os dados usados nos experimentos como conjuntos adicionais que não foram usados pela indisponibilidade de rótulos sobre eventos. O repositório representa uma entrega relevante deste trabalho, uma vez que além da disponibilização para uso pela comunidade científica, foi realizado amplo trabalho de extração, transformação e carga (*Extract, transform and load* (ETL)) para organização e estruturação das séries de maneira padronizada. O tratamento empregado nas séries contribui para simplificar seu uso em processos de análise de séries, especialmente para tarefa de detecção de eventos.

Apesar das séries contidas no repositório DAL *Events Datasets* serem oriundas de bases públicas, a estruturação foi necessária para simplificar o processo de execução de experimentos. Os dados originais estão em formatos não uniformizados como arquivos de texto plano, planilhas eletrônicas, textos separados por vírgulas (csv) e tabelas em diferentes sites da Internet. Portanto, o uso direto das fontes originais demanda um trabalho de preparação exaustivo e passível de falhas.

A estruturação consiste em acessar os dados originais de cada conjunto, criar *scripts* padronizados de carga, organização das séries em objetos da linguagem R reutilizáveis, criar cópias de cada conjunto usando o mesmo tipo de arquivo[4]. Uma vez padronizados os arquivos e os *scripts* de ETL de cada conjunto de dados, foi desenvolvido um pacote na linguagem R que permite que os dados sejam carregados por meio de uma instrução simplificada referenciado o nome da série desejada. Por fim, foi elaborado um relatório como documentação do repositório na plataforma GitHub com informações sobre os conjuntos de dados e instruções sobre forma de acesso aos dados originais e exemplos de carga simplificada por meio do pacote R desenvolvido[5].

O conjunto de dados *Gecco Challenge* contém dados ambientais relacionados ao monitoramento de componentes de água potável na Alemanha para identificar anomalias que indiquem poluição [Moritz et al., 2018]. O conjunto de dados é composto por nove séries com dados reais de componentes da água, com aproximadamente 140 mil linhas. Entretanto, há um intervalo de 1.500 linhas contendo 72 eventos imputado de maneira sintética pelos responsáveis pela publicação do conjunto de dados para uso em avaliação de métodos de detecção de eventos.

As séries contidas nos conjuntos de dados Yahoo *Labs*, NAB e RARE possuem dados oriundos de áreas de domínio ligados à tecnologia, tais como tráfego de dados em rede e internet, computação em nuvem e monitoramento de equipamentos de informática [Ahmad et al., 2017; Lomio et al., 2020]. As séries do Yahoo *Labs* e NAB são extensivamente usadas em pesquisas de comparação de métodos de detecção de eventos [Hasani, 2020; Salles et al., 2020; Lomio et al., 2020]. esse uso extensivo motivou a inclusão desses conjuntos de dados na seleção e uso de todas as séries neles contidas.

[4]Dados originais disponíveis em: https://github.com/cefet-rj-dal/event_datasets/tree/main/data

[5]Relatório e instruções de uso dos conjuntos de dados disponíveis em: https://github.com/cefet-rj-dal/event_datasets#readme

Para maior diversidade de conjuntos de dados relativos à tecnologia, importante área de domínio ao se falar em dados em *streaming*, foi adicionado o conjunto de dados RARE. A publicação do RARE é mais recente que Yahoo *Labs* e NAB e contém dados específicos para estudos de identificação de anomalias em memória de equipamentos responsáveis pelo tráfego de dados em rede [Lomio et al., 2020]. esse conjunto contém mais de 7.000 séries, das quais foram selecionadas 20 séries aleatoriamente para os experimentos.

Os dados do UCR *Anomaly Archive* [Wu and Keogh, 2021] são séries desenvolvidas especialmente para resolver problemas comuns em conjuntos de dados que vem sendo usados como *benchmark* de detecção de eventos. O conjunto de dados UCR abrange muitos domínios, incluindo medicina, esportes, entomologia, indústria, ciência espacial, robótica. Os tipos de falhas apontados em outros conjuntos de dados de *benchmark* são: trivialidade, densidade de anomalia irreal, verdade fundamental mal rotulada e viés de execução até falha.

As falhas que o UCR *Anomaly Archive* busca mitigar podem inclusive contestar parte do avanço da ciência nessa área de pesquisa [Wu and Keogh, 2021]. esse conjunto de dados contém centenas de séries e extensão expressivamente maior que as demais séries selecionadas para os experimentos. Foram selecionadas 20 séries aleatoriamente para os experimentos.

O conjunto de dados *University of California, Irvine* (UCI) 3W possui séries baseadas em dados reais de monitoramento de equipamentos de perfuração de poços de petróleo. Cada série contém dados relativos a diferentes poços com informações sobre pressão, temperatura e fluxo de elevação de gás. Os dados contêm rótulos de eventos raros relacionados a ocorrências indesejáveis que podem gerar perdas ou mesmo acidentes no processo de perfuração.

Além dos dados reais, oriundos de poços perfurados pela empresa Petrobras, há dados sintéticos adicionados no UCI 3W. As características dessas séries tornam complexa a tarefa de detecção de eventos, tendo em vista que em muitos casos os eventos são considerados raros, localizados em pontos únicos ou pequenos intervalos de séries com milhares de observações. Também foram selecionadas 20 séries aleatoriamente para execução dos experimentos.

Nas próximas seções deste capítulo são analisados os resultados dos experimentos e discutidas suas contribuições para a detecção de eventos em séries temporais em *streaming*. Primeiramente, é exibido um caso de exemplo ilustrativo com a análise exploratória de uma série temporal real, ilustrando suas características, eventos existentes e o resultado da execução de métodos usando o *Nexus*. Em seguida, é apresentado o panorama completo dos experimentos com todas as séries e métodos avaliados, incluindo métricas sobre os resultados para cada conjunto de dados e diferentes configurações de execução.

V.2 Exemplo ilustrativo: Análise exploratória e detecções com *Nexus* da série *pH*

A fim de permitir uma compreensão do fluxo completo de análise de uma série temporal usando o *Nexus*, nesta seção é exibida a análise exploratória da série temporal real *pH* do conjunto de dados *Gecco Challenge*. Essa série representa o nível de *pH* da água em um intervalo equivalente a um extrato do conjunto de dados com aproximadamente um dia da série temporal com 1500 observações e 72 eventos rotulados. São apresentadas suas características, eventos existentes e o resultado da execução de métodos usando o *Nexus*.

A Figura V.1 contém a visualização da série *pH* com o comportamento dos dados ao longo do tempo. A distribuição dos dados da série, por sua vez, é analisada visualmente por intermédio da Figura V.2. Mesmo na análise visual direta, é possível observar na evolução dos valores ao longo do tempo que há alguns pontos na série com valores muito acima das demais observações.

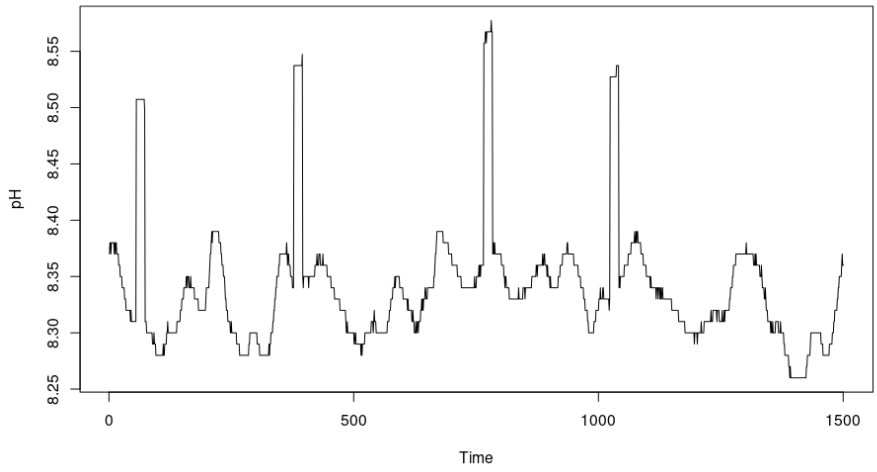


Figura V.1: Dados do pH de água durante um dia com ocorrência de eventos rotulados - Conjunto de Dados *Gecco Challenge*

A presença de valores discrepantes nos dados da série é acentuada por meio da análise do diagrama de caixa da Figura V.2. Os valores discrepantes são identificados no gráfico pelos pontos na região superior, bem afastados do limite superior onde são esperadas a maioria das observações da série. Os rótulos existentes nesse conjunto de dado confirmam que os eventos reais estão na parte superior da série, concentrada nas regiões próximas aos picos com valores de $pH > 8,5$, como é explorado continuidade desta seção.

As Figuras V.3 e V.4 permitem visualizar os resultados das detecções dos métodos FBIAD, ARIMA e CF *offline* e *online*, respectivamente. As detecções foram realizadas por meio do *framework Harbinger* no cenário *offline* e do *Nexus* no cenário *online*. Essa comparação permite explorar a pergunta de pesquisa que questiona se os resultados de métodos *offline* se mantém no

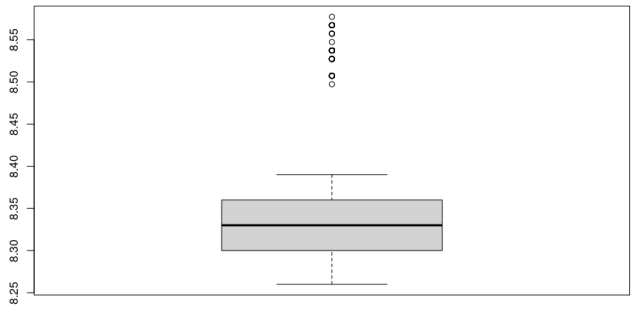


Figura V.2: Análise de distribuição de dados da série pH usando diagrama de caixa

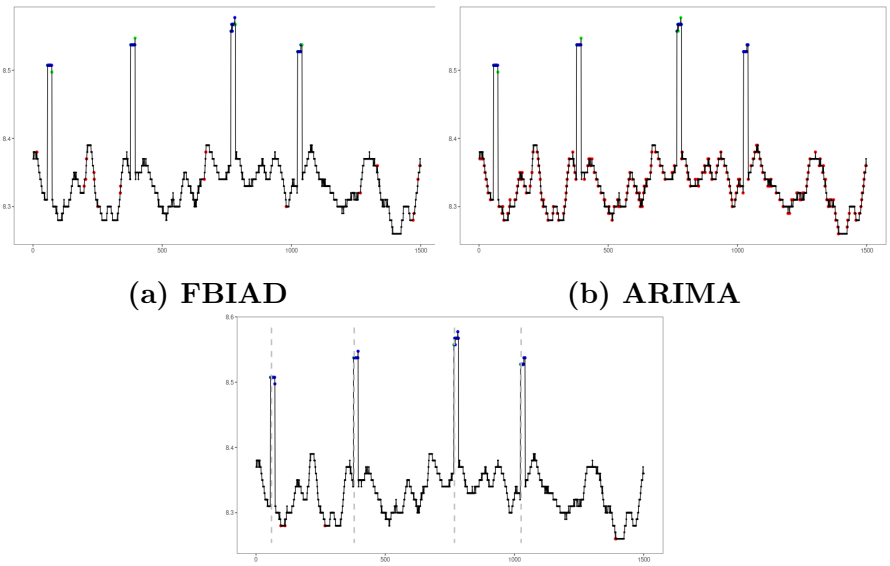
cenário *online*.

Nessa análise inicial *offline*, apesar de existir falsos negativos para o FBIAD, ele consegue marcar eventos na região onde eles ocorrem e há poucos falsos positivos, figura V.3 (a). Isto ocorre porque esse método é focado em identificar anomalias e na detecção *offline* o método tem acesso completo aos dados, o que facilita identificar a distribuição da série completa. Na execução *online*, o método FBIAD marcou todos os eventos reais, o que pode ser visto pela ausência de pontos marcados em azul (falsos negativos) nos resultados desse método. Contudo, há muitos falsos positivos, como pode ser visto pelos pontos em vermelho na Figura V.4 (a).

O método ARIMA apresenta comportamento similar em ambos cenários nessa avaliação inicial. Tanto na detecção *offline* quanto *online* esse método apresenta muitos falsos positivos, como pode ser visto pelos pontos em vermelho nas Figuras V.3 (b) e V.4 (b). esse resultado é interessante, pois mostra que mesmo em um caso básico há indicativos de capacidade do *Nexus* contribuir para adaptação de métodos desenvolvidos inicialmente para o cenário *offline* à detecção em *streaming*.

Na detecção com o método CF há poucos falsos positivos em ambos cenários. Porém, esse método com não identifica nesses exemplos adequadamente todos os eventos, pois a maioria dos pontos com eventos reais estão marcados como falsos negativos. É importante atentar para o fato desse ser um método especializado na detecção de pontos de mudança, o que influencia nesses resultados.

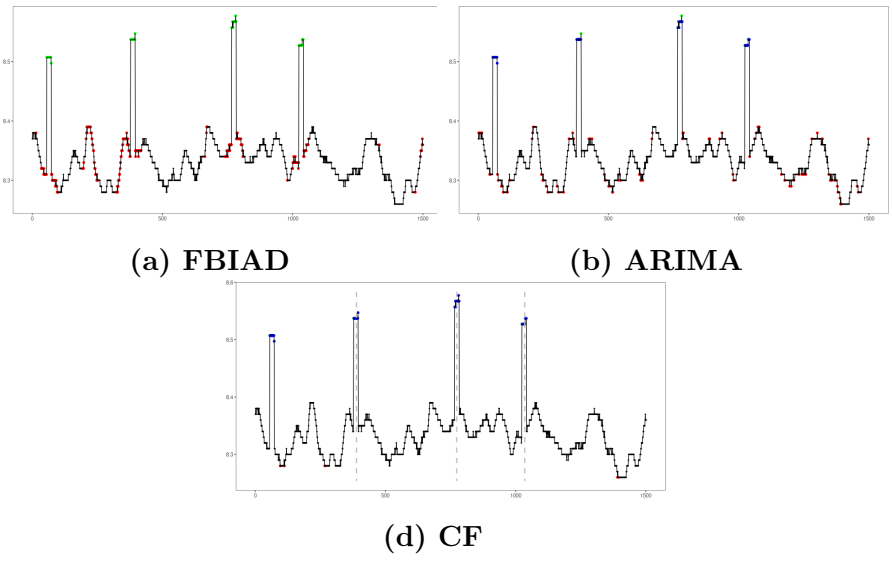
As marcações verticais com linhas tracejadas nas Figuras V.3 (d) e V.4 (d) indicam os pontos onde o método detectou mudanças. Se forem considerados como pontos de mudança os picos da série onde há eventos, percebe-se que em todos os casos *offline* o método detectou corretamente o início de mudança na série. Na detecção *online* não foi possível detectar o primeiro ponto de mudança, mas os demais foram identificados corretamente. esse tipo de situação tem relação com a maior complexidade da análise em *streaming*, pois o método tem acesso a menos informações para aprender sobre a distribuição dos dados no começo da série.



(a) FBIAD **(b) ARIMA**

(d) CF

Figura V.3: **Nexus - Detecção *offline* série pH - Comparação de métodos**. Comparação de métodos.



(a) FBIAD **(b) ARIMA**

(d) CF

Figura V.4: **Nexus - Detecção *online* série pH**. Comparação de métodos. **Valores de parâmetros**: $w = 243, s = 243$, $m = 0 \rightarrow$ Memória completa, $P(e) > 0 \rightarrow$ sem filtro de probabilidade

A análise sem filtros de probabilidade e sem testes de outras configurações de parâmetros do *Nexus*, como a configuração que gerou as visualizações da Figura V.4 representa os resultados iniciais das detecções *online*. A seguir aprofunda-se tanto essa primeira configuração e os resultados das métricas (Tabela V.4) quanto outras configurações de tamanho de memória e análise da probabilidade dos eventos $P(e)$. A primeira variação de configurações para comparação é visualizada nas Figuras V.5 e V.6, onde se comparam as detecções com o método FBIAD com memória completa e memória parcial de três lotes.

Ao executar o método FBIAD com memória parcial dos três lotes mais recentes ao invés da

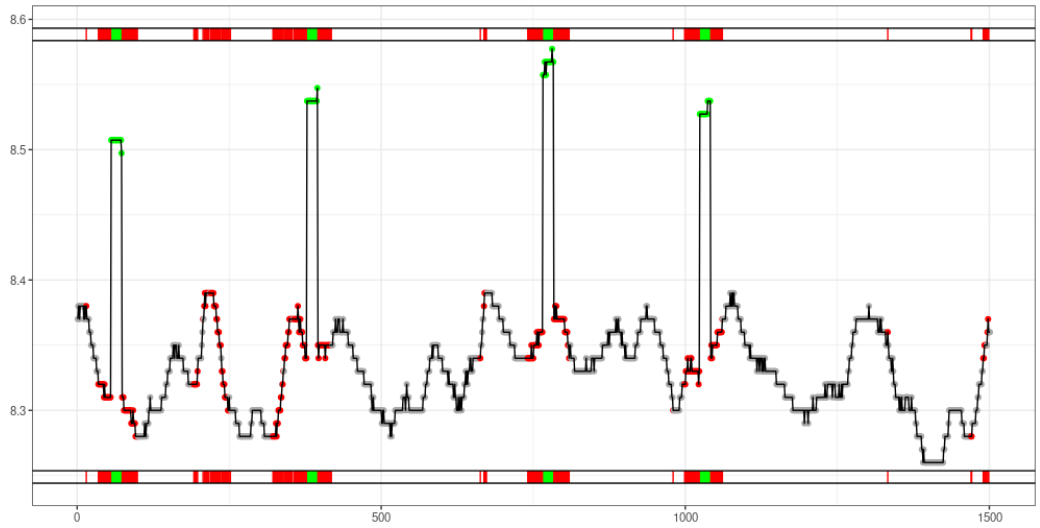


Figura V.5: **Nexus Série pH (FBIAD) - Comparação memória de lotes - Completa - Valores de parâmetros**: $w = 243, s = 243, m = 0 \rightarrow$ Memória completa

memória completa, nota-se por meio dos pontos em azul na Figura V.6 diversos eventos concentrados em uma região da série não foram identificados. Essa situação está relacionada ao fato de existir menos dados em cada iteração do *streaming* e é esperado pela intuição sobre a memória de lotes detalhada na metodologia no Capítulo IV. esse tipo de exemplo é importante para reforçar a necessidade de comparação de diferentes configurações, bem como da análise conjunta de valores tabulados de métricas com a análise visual.

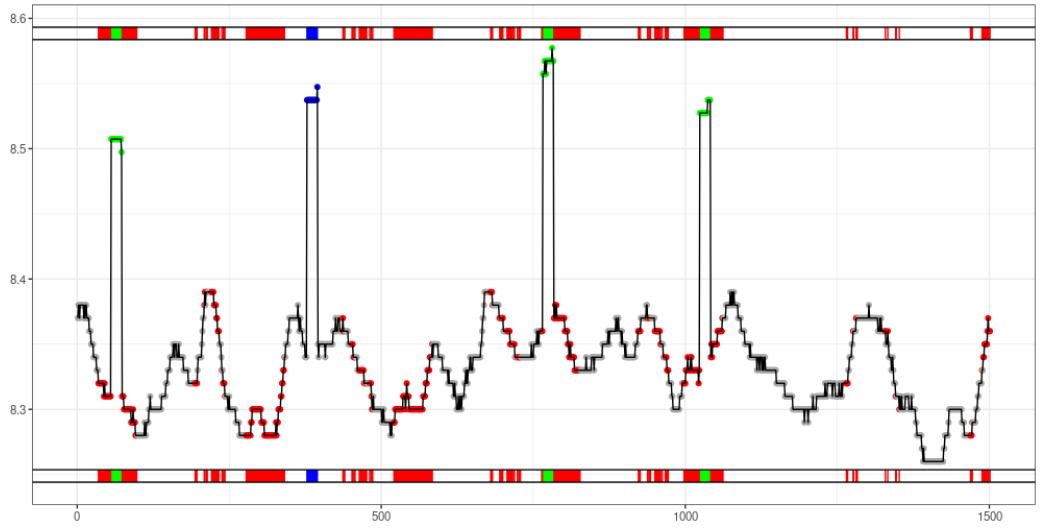


Figura V.6: **Nexus Série pH (FBIAD) - Comparação memória de lotes - Parcial - Valores de parâmetros**: $w = 243, s = 243, m = 3 \rightarrow$ Memória parcial de três lotes

A exploração de resultados por meio das métricas abordadas mais adiante aprofunda o entendimento dos resultados. Deve-se atentar inclusive quanto ao equilíbrio entre eventuais falsos positivos, seus impactos na acurácia dos métodos e o tempo de execução, cuja simples visualização não con-

segue revelar. Antes de prosseguir para análise de métricas, são vistos a seguir outros aspectos que a análise visual revela como efeito, redução de falsos positivos e velocidade de execução que a memória.

As Figuras V.7 e V.8 apresentam a análise de um extrato com 500 observações da série *pH*, possibilitando uma visualização detalhada dos primeiros conjuntos de eventos presentes na série. Esse exemplo ajuda a entender visualmente o uso da probabilidade de eventos $P(e)$ como uma contribuição para detecção de eventos em séries em *streaming*. Nas Figuras V.7 e V.8 são apresentados os resultados das detecções com marcações de eventos com $P(e) > 0,5$ e $P(e) > 0,8$, respectivamente. Esses valores de probabilidade representam pontos marcados como eventos na maioria simples das vezes para $P(e) > 0,5$ ou com base na lógica de análise de Pareto, representando as marcações em 80% dos casos para $P(e) > 0,8$.

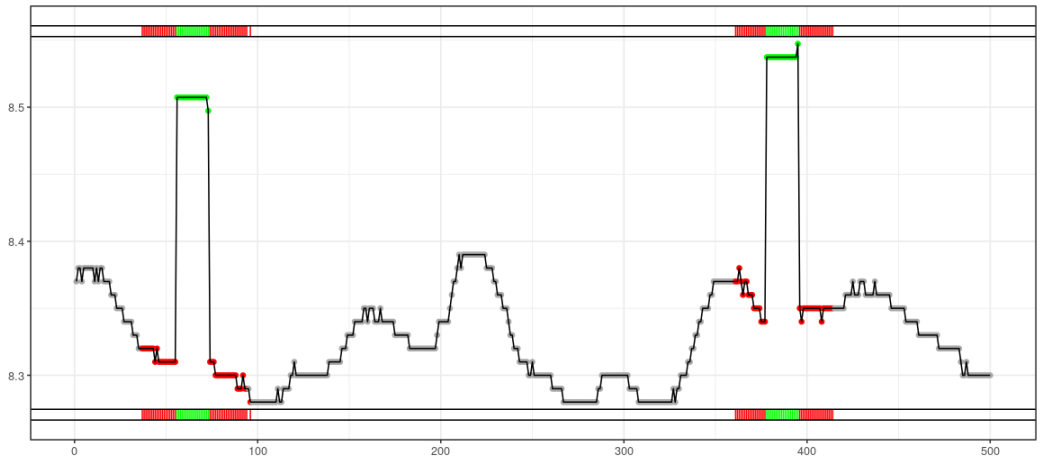


Figura V.7: **Nexus Série pH (primeiras 500 observações)** - Detecção usando memória completa, exemplos de filtro de valor de $P(e)$. **Valores de parâmetros**: $w = 81, s = 81$, $m = 0 \rightarrow$ Memória completa. **Opções de filtro**: $P(e) > 0,5 \rightarrow$ Pontos marcados como eventos na maioria simples das vezes.

No exemplo da Figura V.7, percebe-se na configuração $P(e) > 0,5$ que os eventos reais (marcados em verde) foram identificados adequadamente. Contudo, o método também identificou falsos positivos (marcados em vermelho) nas regiões ao redor dos eventos reais. Por outro lado, a Figura V.8 revela que aumentando o rigor do filtro para $P(e) > 0,8$, o resultado restante mantém os eventos reais e reduz os falsos positivos. Esses resultados apoiam a hipótese de impactos positivos pelo uso de lotes na detecção de eventos em séries em *streaming*, principalmente com a associação de análise de probabilidade.

Além de efeitos na acurácia, esses resultados também ajudam a corroborar a hipótese que o uso de memória completa ou parcial é importante para o tempo de execução e a capacidade de detecção precoce das ocorrências indesejáveis originadas por eventos. A intuição apresentada na metodologia do *Nexus* (Capítulo IV, Seção IV.3) é que a memória parcial tende a reduzir tanto o tempo total

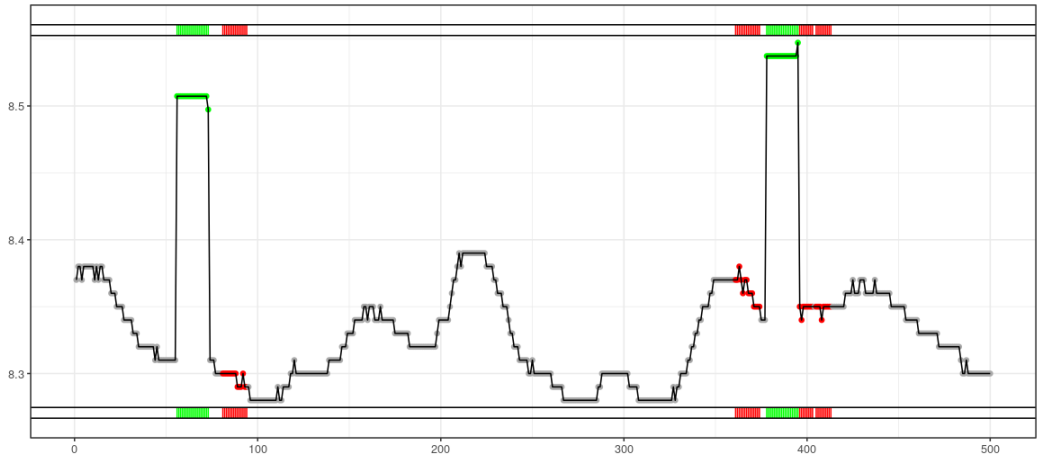


Figura V.8: **Nexus Série pH (primeiras 500 observações)** - Detecção usando memória completa, exemplos de filtro de valor de $P(e)$. **Valores de parâmetros**: $w = 81, s = 81$, $m = 0 \rightarrow$ Memória completa. **Opções de filtro**: $P(e) > 0,8 \rightarrow$ Pontos marcados como eventos em 80% dos casos (Pareto).

quanto o tempo de processamento de cada lote no *streaming*. Apesar do tamanho reduzido da série pH, as Figuras V.9 e V.10 colaboram com a confirmação dos impactos positivos na redução do tempo de execução dos métodos ao usar memória parcial.

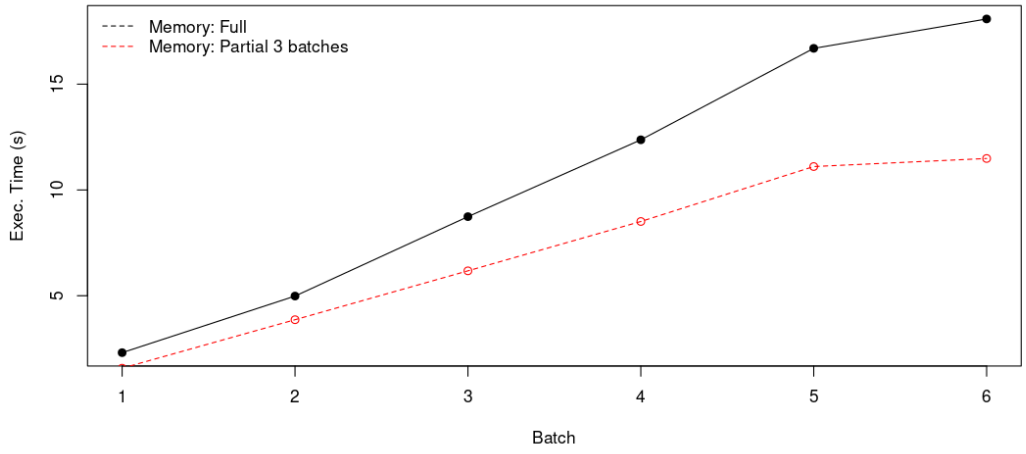


Figura V.9: **Tempo de Execução Acumulado (FBIAD)** - Tempo de execução acumulado ao longo dos lotes da série pH. **Valores de parâmetros**: $w = 243, s = 243$, $m = 0 \rightarrow$ Memória completa e $m = 3 \rightarrow$ Memória parcial.

Ambos exemplos de comparação do tempo de execução por lote ilustram que os impactos na velocidade de tempo de resposta são relevantes. Diante dessa informação, é aconselhável que a acurácia dos métodos seja avaliada de maneira integrada à análise de tempo, dada a possibilidade de crescimento contínuo do volume de dados em séries em *streaming*. Por isso, na comparação dos resultados foi incluída a métrica tempo médio por lote (TML).

Na Tabela V.4 são apresentados os resultados das métricas para a variável pH, cujos melhores

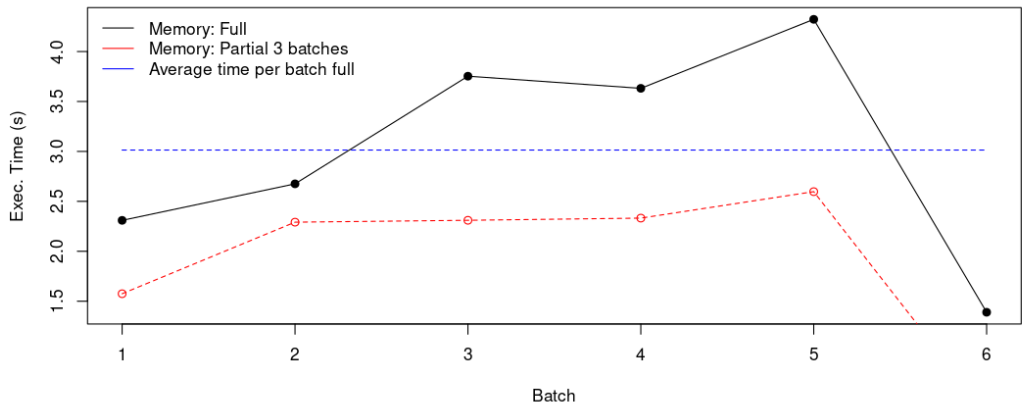


Figura V.10: **Tempo de Execução por Lote (FBIAD)** - Tempo de execução para cada lote da série pH. **Valores de parâmetros**: $w = 243, s = 243$, $m = 0 \rightarrow$ Memória completa e $m = 3 \rightarrow$ Memória parcial.

resultados estão com os valores sublinhados na célula equivalente ao respectivo método. Para avaliar de maneira adequada o comportamento dos métodos quanto à detecção correta e velocidade, as métricas usadas são: acurácia, F_1, F_1 *SoftED* e TML em segundos. Além disso, essas métricas são apresentadas para execuções usando memória completa ou memória de três lotes da série.

Tabela V.4: **Nexus para série pH** - Comparação de métodos e impacto de memória de lotes. **Valores de parâmetros**: $w = 243$, $s = (27, 243)$, $m = (0, 3)$. **Legenda Opções de Memória**: $m = 0 \rightarrow$ Memória completa, $m \geq 1 \rightarrow$ Memória parcial. **Métricas**: Acurácia, F_1, F_1 *SoftED* e tempo médio por lote (TML) em segundos (seg.).

Métrica	**Método**	$s = 243$		$s = 27$
		$m = 0$	$m = 3$	$m = 0$
Acurácia	FBIAD	0,81	0,72	0,81
	ARIMA	0,90	0,89	0,82
	CF	0,95	0,95	0,95
F_1	FBIAD	0,34	0,20	0,34
	ARIMA	0,11	0,02	0,06
	CF	0,17	0,05	0,17
F_1 ***SoftED***	FBIAD	0,34	0,20	0,34
	ARIMA	0,14	0,06	0,08
	CF	0,17	0,07	0,17
TML (seg.)	FBIAD	3,15	1,98	0,38
	ARIMA	1,30	0,99	0,35
	CF	1,83	1,21	0,22

A acurácia oferece uma visão geral dos acertos tanto da existência de eventos quanto da inexistência. A métrica F_1 permite avaliar se um método obteve resultados equilibrados, pois apresenta valores mais elevados para métodos com muitos acertos e poucos falsos positivos [Han et al., 2012]. Enquanto a F_1 *SoftED* é uma métrica pertencente a um conjunto de métricas que suavizam os efei-

tos de métricas[6] de classificação tradicional para incorporar tolerância na imprecisão de detecção de eventos [Salles et al., 2023b].

Ao avaliar a acurácia dos métodos, percebe-se que nas execuções com memória completa o método CF apresenta resultados superiores com 95% de acurácia, enquanto o FBIAD tem os menores valores. O impacto da memória de três lotes ou completa foi percebido com relevância apenas no método FBIAD com uma redução de $-11,1\%$ na acurácia. No caso do método CF não houve qualquer alteração, inclusive mantendo esse como a maior acurácia nas detecções com memória de três lotes e nos demais métodos houve reduções abaixo de -5%.

Dados os resultados da Tabela V.4, para uma análise do equilíbrio entre a detecção correta dos eventos e a velocidade de execução, pode-se avaliar conjuntamente os resultados das métricas acurácia, F_1 e F_1 *SoftED* comparando-as com o TML. Enquanto para F_1 e F_1 *SoftED* o método FBIAD apresenta os melhores resultados, o método *CF* se destaca na acurácia. Em relação ao TML, o método ARIMA mostra ser o mais rápido, mas os valores são próximos quando as execuções usam lotes menores ou diminui o tamanho de memória.

Os achados da avaliação dos experimentos até este ponto indicam que o *Nexus* contribui para que mesmo métodos criados para detecção *offline* continuam válidos na detecção *online*. Métricas de acurácias, F_1 e F_1 *SoftED* relativa ao acerto de detecções não sofreram mudanças significativas em diferentes condições de memória. O uso de lotes com diferentes tamanhos e configurações de memória influenciam no tempo de processamento que impacta o custo computacional e a detecção precoce.

Para ampliar a análise da velocidade dos métodos, a Figura V.11 apresenta a distribuição detalhada dos tempos por lote que deram origem aos valores de TML para os três métodos, considerado o cenário base com $w = 243, s = 243, m = 0$. Na avaliação apenas da métrica TML o método ARIMA tem o melhor resultado, contudo, como os métodos FBIAD e CF obtiveram melhores resultados nas outras métricas, antes de considerar o valor da métrica gerada a partir da média de tempos como resultado definitivo, cabe a análise da distribuição e realização de testes estatísticos complementares para confirmar se efetivamente há significância estatística na diferença entre os resultados dos métodos.

A análise gráfica simplificada do diagrama de caixa da Figura V.11 parece mostrar que efetivamente os valores dos três métodos tem diferenças, porém com regiões de resultados similares, principalmente no caso do ARIMA e CF. Para confirmar essa assertiva, o teste estatístico *t.test* pareado, com intervalo de confiança de 95% e *valor-p*$> 0,05$, indicou que há diferença com significância apenas na comparação entre os valores do FBIAD em relação aos outros dois métodos. Contudo, não há diferença nos tempos de execução por lote entre os métodos ARIMA e CF.

[6]Disponível em: https://github.com/cefet-rj-dal/softed

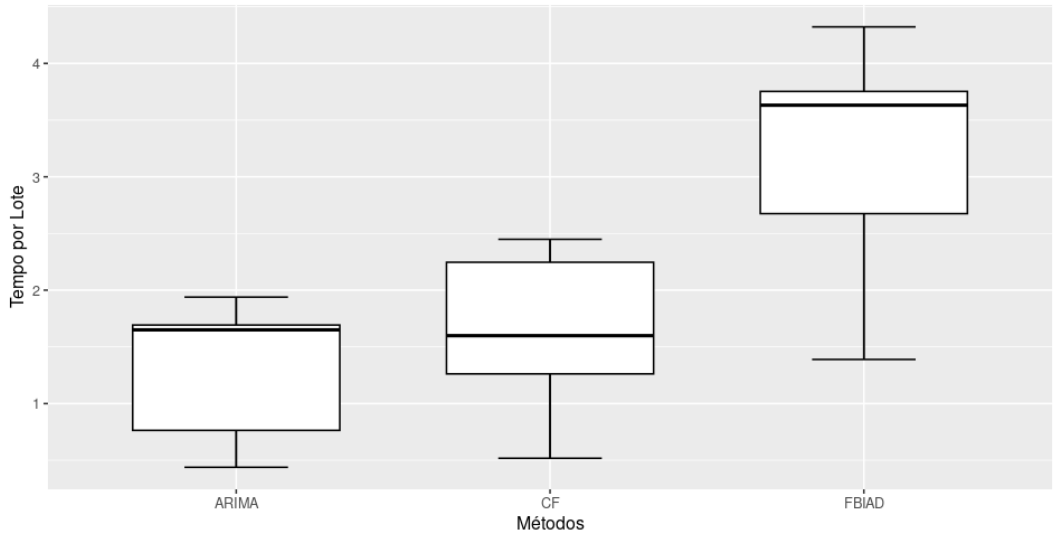


Figura V.11: **Tempo de Execução por Lote para série pH - *Métodos ARIMA, CF, FBIAD*. Valores de parâmetros**: $w = 243, s = 243, m = 0$

Para analisar como os métodos se comportam no *streaming* por meio do *Nexus* é importante avaliar o *Lag* das detecções. O *Lag* identifica quantos lotes o método demorou até detectar um evento, ajudando a identificar a capacidade de detecção precoce de eventos dos métodos. Seu cálculo usa a Equação IV.3 que compara a primeira vez em que um evento no ponto t é lido (sb_t) com a primeira vez em que o método o detecta (fdb_t).

A análise de *Lag* de detecções para a série pH é exibida na Tabela V.5, onde consta apenas os métodos que detectaram eventos para cada ponto t avaliado. Uma vez que o *lag* é calculado para cada momento no tempo da série, foram obtidos por meio de seleção aleatória de 2 valores de t com presença de eventos, além de incluir na seleção o primeiro e último evento na série $(56, 1041)$. Assim, os momentos t selecionados aleatoriamente da série são 67 e 770 e a seleção final adicionou os momentos t 56 e 1041, com o primeiro e último evento.

Tabela V.5: **Nexus para série pH - *Lag*** - Análise de *Lag* de detecções. **Cálculo do *Lag***: $Lag_t = fdb_t - sb_t$ (veja Equação IV.3). **Variáveis do cálculo**:$fdb_t \rightarrow$ Primeira detecção de evento no ponto t, $sb_t \rightarrow$ Primeiro lote em que o ponto t foi lido. **Posições dos eventos analisados**: Seleção aleatória (67, 770), adicionados primeiro e último evento na série (56, 1041). **Parâmetros do Nexus**: $w, s = 243$, $m = (0, 3)$

Posição t do evento	**Método**		$m = 0$		$m = 3$	
		sb_t	fdb_t	Lag_i^s	fdb_t	Lag_i^s
Posição $t = 56$	FBIAD	1	1	0	1	0
	ARIMA		1	0	1	0
	CF		1	0	1	0
Posição $t = 67$	FBIAD	1	1	0	1	0
Posição $t = 770$	FBIAD	2	3	1	3	1
Posição $t = 1041$	FBIAD	3	4	1	4	1

Nos resultados para a seleção de eventos analisados com $w = 243, s = 243$ e memória $m = 0$

ou $m = 3$, nota-se uma predominância do método FBIAD, que detectou os quatro eventos e obteve resultados iguais aos demais métodos no caso de $t = 56$ em que todos detectaram o evento. Não houve diferenças de *lag* entre as opções de memória. Além disso, em muitos casos alguns métodos não detectaram o evento no ponto selecionado aleatoriamente para análise.

No primeiro evento da série, $t = 56$ todos os métodos não apresentam atraso na detecção, ou seja, tem $Lag_{56} = 0$. Para os demais eventos nos momentos apenas o FBIAD realizou a detecção corretamente, com $Lag_{67} = 0$, $Lag_{770} = 1$ e $Lag_{1041} = 1$. Dados esses resultados, para aprofundar o entendimento do método FBIAD e avaliar o Lag mediano da série pH para $w = 243, s = 243, m = 0$ é possível realizar a análise visual por meio da Figura V.12.

Na Figura V.12 observa-se que o valor da mediana é $Lag = 1$, porém, a visualização permite verificar mudanças de comportamento ao longo do *streaming*. No caso dessa série, o primeiro evento detectado e alguns na primeira metade das detecções estão bem acima da mediana, chegando próximo de $Lag_t = 4$. Contudo, no geral, a maioria dos eventos tem Lag próximo a um ou não tem qualquer atraso ($Lag_t = 0$).

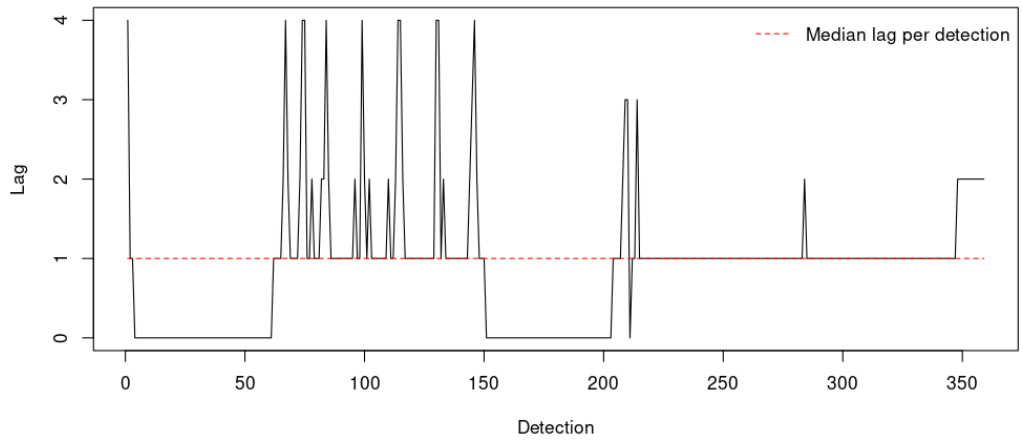


Figura V.12: **Análise visual de *Lag***: - Método FBIAD, série pH - Análise completa do valor mediano de *Lag*. **Parâmetros do Nexus**: $w, s = 243$, $m = 0$

A série pH estudada na análise exploratória desta seção possui limitações que podem levar a situações muito simples ou complexas, dependendo das configurações dos parâmetros dos métodos ou do *Nexus*. Por um lado, a grande diferença da região dos valores dos eventos reais em relação à distribuição dos demais dados da série pode tornar a detecção um desafio simples para alguns métodos. Por outro lado, a existência de poucas observações limita as possibilidades de tamanhos de lotes, bem como opções de memória parcial e impacta métodos que precisam de mais dados para treinamento. A próxima seção apresenta os resultados completos em diversidade maior de séries para aprofundar as possibilidades de uso e contribuições do *Nexus*.

V.3 Análise do panorama completo de resultados

Esta seção visa aprofundar os achados do experimento simplificado que corroboram a hipótese de que o uso de lotes deslizantes contribui para detecção precoce de eventos em *streaming* e redução de custo computacional mantendo os níveis acurácia. Para tanto, são apresentados resultados mais abrangentes em séries com comportamento e tipos de eventos distintos. Os conjuntos de dados usados representam várias áreas de domínio e diferentes complexidades de detecção, inclusive pelo volume de dados presentes em cada série.

Para encadeamento lógico da análise, a avaliação foi divida em três subseções. A Subseção V.3.1 apresenta os resultados das detecções com foco nas métricas de acurácias e discussão dos conjuntos de dados. A Subseção V.3.2 avalia aspectos alusivos ao tempo de processamento, impacto das configurações do *Nexus* no custo computacional e influência na detecção precoce de eventos. A Subseção V.3.3 conclui a avaliação aprofundando a discussão sobre a detecção precoce de eventos e como a análise de *Lag* contribui para esse aspecto do comportamento de métodos em *streaming*.

V.3.1 Detecções, acurácia e impacto de lotes deslizantes na execução dos métodos

Esta subseção apresenta o resumo dos resultados das detecções. Também é discutido o impacto de lotes deslizantes na execução dos métodos. Além de fornecer uma visão geral que expande o exemplo da Seção V.2, os resultados das detecções são explorados nas próximas subseções à luz do detalhamento da hipótese e perguntas de pesquisa deste trabalho.

Os primeiros conjuntos de dados cujos resultados são discutidos nas Tabelas V.6, V.7 e V.8 são Yahoo *Labs*, NAB e RARE. Esses conjuntos contêm séries relacionadas a tecnologia como monitoramento de redes, tráfego de dados na internet e monitoramento de equipamentos usados em redes e plataformas de nuvem. Conforme mencionado em seções anteriores, Yahoo *Labs* e NAB são amplamente usados como *benchmark*, enquanto RARE tem publicação mais recente e foca no monitoramento de métricas relacionadas ao uso de memória.

Um comportamento percebido na Tabela V.6 para o método CF é a necessidade de *warmup* maior que os outros métodos. Isso pode ser visto tanto na avaliação por grupo quanto nos resultados consolidados do Yahoo *Labs*. Essa informação é confirmada pelos valores NA nas métricas com *warmup* de tamanho $w = 81$, enquanto o CF executou normalmente mesmo com lotes menores como $s = (27, 81)$ quando se aumenta *warmup* para $w = 243$. A avaliação desse tipo de mudança no comportamento com alteração de parâmetros do *Nexus* contribui para confirmar que a forma de execução dos métodos influencia seus resultados em *streaming*.

O CF manteve o comportamento similar de necessidade de *warmup* maiores para conseguir detectar eventos nas séries NAB e RARE, como se vê nas Tabelas V.7 e V.8. Entretanto, ao alterar

Tabela V.6: **Nexus Conjunto de dados YAHOO LABS consolidado** - Comparação de métodos quanto ao tamanho de *warmup* (w), tamanho de lotes (s) e impacto de memória de lotes (m). **Valores de parâmetros**: $w = (243, 81), s = (243, 81, 27)$; **Opções de Memória**: $m = 0 \rightarrow$ Memória completa, $m \geq 1 \rightarrow$ Memória parcial. **Métricas**: Média de acurácia, F_1, F_1 *SoftED* por série e tempo médio por lote (TML) em segundos (seg.).

Métrica	**Método**	$w = 243, s = 243$		$w = 81, s = 81$		$w = 243, s = 27$
		$m = 0$	$m = 3$	$m = 0$	$m = 3$	$m = 0$
Acurácia	FBIAD	0,91	0,92	0,91	0,94	0,91
	ARIMA	0,93	0,93	0,93	0,93	0,93
	GARCH	0,97	0,97	0,97	0,98	0,98
	CF	0,97	0,97	NA	NA	0,97
	LSTM	0,93	0,94	0,98	0,95	0,93
F_1	FBIAD	0,14	0,13	0,13	0,08	0,17
	ARIMA	0,11	0,07	0,11	0,05	0,11
	GARCH	0,12	0,11	0,13	0,06	0,18
	CF	0,19	0,14	NA	N A	0,20
	LSTM	0,09	0,08	0,09	0,05	0,09
F_1 ***SoftED***	FBIAD	0,15	0,15	0,14	0,10	0,18
	ARIMA	0,12	0,10	0,13	0,08	0,13
	GARCH	0,35	0,22	0,33	0,13	0,35
	CF	0,22	0,18	NA	NA	0,23
	LSTM	0,11	0,10	0,10	0,09	0,10
TML (seg.)	FBIAD	4,01	2,46	0,98	0,40	0,35
	ARIMA	12,97	8,15	3,58	1,65	1,51
	GARCH	66,98	52,64	18,01	13,92	5,99
	CF	2,45	1,42	NA	NA	0,20
	LSTM	11,38	10,53	4,28	5,17	2,40

o tamanho do *warmup* para $w = 243$ o desempenho passa a ser similar a outros métodos. Em alguns casos, como no Yahoo *Labs* na configuração $w = 243, s = 243, m = 0$, esse método tem resultados até superiores em muitas métricas.

As métricas relacionadas às detecções corretas são usadas para análise dos métodos ao longo do *streaming*. Mas a análise tem foco maior em verificar se um método que apresenta determinado nível de acurácia, F_1 ou F_1 *SoftED* mantém níveis similares em diferentes configurações de parâmetros do *Nexus*. No geral, ao avaliar essas métricas nas Tabelas V.6 ,V.7 e V.8 não é possível notar mudanças severas nas diferentes configurações de execução do *Nexus*.

Para avaliar o comportamento do *Nexus* em outras áreas de domínio, a Tabela V.9 apresenta os resultados para os conjuntos de dados UCI 3W *OIL WELLS* e UCR *Anomaly Archive*. Conforme apresentado nas configurações dos experimentos da Seção V.1 o UCI 3W contém séries sobre a exploração de petróleo com rótulos relacionadas a eventos raros, inclusive com dados reais da empresa Petrobras e o UCR *Anomaly Archive* tem principalmente dados médicos com eventos sintéticos. As diferenças de tempo de execução entre memória completa e parcial foi amplamente explorada nas etapas anteriores deste capítulo, por isso para esses conjuntos de dados explora-se a diferença entre

Tabela V.7: **Nexus Conjunto de dados NAB** - Comparação de métodos quanto ao tamanho de *warmup* (w), tamanho de lotes (s) e impacto de memória de lotes (m). **Valores de parâmetros**: $w = (243, 81), s = (243, 81, 27)$; **Opções de Memória**: $m = 0 \rightarrow$ Memória completa, $m \geq 1 \rightarrow$ Memória parcial. **Métricas**: Média de acurácia, F_1, F_1 *SoftED* por série e tempo médio por lote (TML) em segundos (seg.).

Métrica	**Método**	$w = 243, s = 243$		$w = 81, s = 81$		$w = 243, s = 27$
		$m = 0$	$m = 3$	$m = 0$	$m = 3$	$m = 0$
Acurácia	FBIAD	0,82	0,79	0,82	0,83	0,82
	ARIMA	0,93	0,91	0,93	0,92	0,93
	GARCH	0,97	0,97	0,98	0,98	0,98
	CF	0,96	0,95	NA	NA	0,96
	LSTM	0,92	0,92	0,92	0,95	0,92
F_1	FBIAD	0,01	0,01	0,01	0,01	0,01
	ARIMA	0,02	0,03	0,02	0,01	0,02
	GARCH	0,03	0,02	0,03	0,01	0,03
	CF	0,02	0,03	NA	NA	0,02
	LSTM	0,01	0,01	0,01	0,01	0,01
F_1 ***SoftED***	FBIAD	0,01	0,01	0,01	0,01	0,01
	ARIMA	0,02	0,02	0,02	0,01	0,02
	GARCH	0,06	0,06	0,06	0,09	0,08
	CF	0,03	0,02	NA	NA	0,03
	LSTM	0,01	0,01	0,01	0,02	0,01
TML (seg.)	FBIAD	23,44	2,56	7,01	0,49	2,50
	ARIMA	73,68	5,77	26,47	0,91	9,59
	GARCH	149,65	33,80	52,68	9,96	15,70
	CF	37,56	1,50	NA	NA	4,59
	LSTM	13,29	14,26	6,96	7,57	4,20

tamanhos de lote.

Para o conjunto de dados UCI 3W os métodos apresentam resultados muito baixos na acurácia com valores máximo de $0,51$ e $0,56$ para o FBIAD com lotes de tamanhos $s = 81$ e $s = 243$, respectivamente. O FBIAD também apresenta resultados melhores de F_1, inclusive com valores mais elevados que os apresentados pelo mesmo método para os demais conjuntos de dados desta seção. Observa-se novamente que o método CF não consegue detectar eventos com lotes $s = 81$ e mesmo com lotes maiores obteve valores muito baixos em todas as métricas de acertos de detecções.

As séries do UCR *Anomaly Archive*, representam o conjunto de dados mais complexo dos experimentos realizados. Isto ocorre porque as séries foram tratadas sinteticamente para conter apenas eventos muito raros, localizadas em pontos específicos no tempo ou em pequenos intervalos das séries [Wu and Keogh, 2021]. Além disso, tem um volume muito superior aos demais conjuntos de dados. Mesmo em um cenário *offline*, o tipo de comportamento das séries do UCR *Anomaly Archive* apresentaria desafios adicionais, pois exige métodos que diferenciem anomalias triviais de eventos raros.

As acurácias elevadas na Tabela V.9 para todos os métodos para UCR *Anomaly Archive* são

Tabela V.8: **Nexus Conjunto de dados RARE** - Comparação de métodos quanto ao tamanho de *warmup* (w), tamanho de lotes (s) e impacto de memória de lotes (m). **Valores de parâmetros**: w, s tamanhos 243 e 81; **Opções de Memória**: $m = 0 \rightarrow$ Memória completa, $m \geq 1 \rightarrow$ Memória parcial. **Métricas**: Média de acurácia, F_1, F_1 *SoftED* por série e tempo médio por lote (TML) em segundos.

Métrica	**Método**	$w = 243, s = 243$		$w = 81, s = 81$		$w = 243, s = 27$
		$m = 0$	$m = 9$	$m = 0$	$m = 9$	$m = 0$
Acurácia	FBIAD	0,93	0,93	0,93	0,94	0,93
	ARIMA	0,91	0,93	0,91	0,93	0,91
	GARCH	0,94	0,94	0,94	0,91	0,94
	CF	0,94	0,94	NA	NA	0,94
	LSTM	0,91	0,91	0,91	0,93	0,91
F_1	FBIAD	0,06	0,09	0,06	0,07	0,06
	ARIMA	0,08	0,07	0,08	0,04	0,08
	GARCH	NA	NA	NA	0,04	NA
	CF	0,02	0,01	NA	NA	0,02
	LSTM	0,04	0,04	0,04	0,04	0,04
F_1 ***SoftED***	FBIAD	0,06	0,10	0,06	0,04	0,07
	ARIMA	0,09	0,10	0,09	0,06	0,09
	GARCH	0,01	NA	0,01	0,08	0,01
	CF	0,04	0,02	NA	NA	0,04
	LSTM	0,06	0,06	0,05	0,08	0,05
TML (seg.)	FBIAD	17,29	5,38	5,79	0,97	1,91
	ARIMA	5,83	3,56	1,84	0,80	0,69
	GARCH	130,69	51,60	41,97	13,04	15,26
	CF	21,81	3,27	NA	NA	2,41
	LSTM	11,49	10,42	4,60	4,31	3,21

resultados da marcação de muitos eventos na série. Porém, como esperado devido à complexidade e tipos de eventos raros nas séries, os métodos estudados não apresentam bons resultados para F_1 e F_1 *SoftED*, pois muitos eventos marcados são falsos positivos. Os desafios propostos por Wu and Keogh [2021] para desenvolvimento de métodos que lidem com o tipo de evento dessas séries fogem do escopo do trabalho atual, mas abre espaço para trabalhos futuros.

V.3.2 Tempo de Processamento

Esta subseção discute aspectos relacionados ao tempo de processamento. São avaliados os achados dos experimentos sobre como diferentes tamanhos de lotes e memória impactam o custo computacional. Esses pontos são relevantes para explorar as questões de pesquisa sobre impacto do processo em *streaming* no tempo de execução de métodos, possibilidade de aplicação de métodos com baixo custo computacional sem perder acurácia e capacidade de detecção precoce dos métodos.

Um primeiro ponto relevante é que tanto a redução de tamanho de lotes quanto de memória reduziu significativamente em todos os casos o tempo de execução. Considerando-se que o processamento em *streaming* pode ocorrer de maneira interrupta, o TML é mais importante que avaliação

Tabela V.9: **Nexus Conjuntos de dados UCR ANOMALY ARCHIVE e UCI 3W OIL WELLS** - Comparação de métodos quanto ao tamanho de *warmup* (w), tamanho de lotes (s) e impacto de memória de lotes (m). **Valores de parâmetros**: $w = (243, 81), s = (243, 81), m = 9$; **Métricas**: Média de acurácia, F_1, F_1 *SoftED* por série e tempo médio por lote (TML) em segundos (seg.).

Métrica	**Método**	**UCI 3W**		**UCR**	
		$w = 81, s = 81$	$w = 243, s = 243$	$w = 81, s = 81$	$w = 243, s = 243$
Acurácia	FBIAD	0,33	0,56	0,90	0,86
	ARIMA	0,26	0,26	0,94	0,94
	GARCH	0,51	0,51	0,97	0,97
	CF	NA	0,24	NA	0,98
	LSTM	0,26	0,27	0,94	0,87
F_1	FBIAD	0,25	0,19	0,01	0,01
	ARIMA	0,09	0,10	0,01	0,01
	GARCH	0,01	0	0,01	0,01
	CF	NA	0	NA	0,01
	LSTM	0,06	0,08	0,01	0,01
F_1 ***SoftED***	FBIAD	0,09	0,08	0,01	0
	ARIMA	0,13	0,06	0,01	0,01
	GARCH	0,01	0	0,01	0,01
	CF	NA	0	NA	0,01
	LSTM	0,06	0,08	0,01	0,01
TML (seg.)	FBIAD	0,98	6,23	1,16	12,12
	ARIMA	0,97	6,86	4,30	38,90
	GARCH	31,80	43,22	10,28	46,28
	CF	NA	3,64	NA	4,41
	LSTM	4,02	9,98	5,01	9,71

de tempo total. Nesse aspecto, nota-se nas Tabelas V.6, V.7 e V.8 em todos os casos o uso de memória parcial confirma intuição de menor tempo de processamento mesmo com lotes tamanhos iguais.

Os achados contribuem também para avaliar a pergunta de pesquisa que questiona a possibilidade de aplicação de métodos com menor custo computacional sem perda na acurácia das detecções. Nota-se em todos os conjuntos de dados deste capítulo que não há uma relação direta entre a simplicidade e baixo tempo de processamento e redução da acurácia das detecções, reforçando os resultados do exemplo ilustrativo. Nota-se, por exemplo, que os métodos FBIAD, CF e ARIMA tem tempos de processamento baixo na série RARE, UCI 3W e não tem acurácia de maneira geral menor. No Yahoo *Labs*, por exemplo, o CF é um dos métodos com menor TML e tem bons resultados nas métricas acurácia e F_1, sendo superior em alguns casos desse conjunto de dados.

Nas séries avaliadas na Tabela V.9 são apresentados dados dos experimentos com memória parcial com tamanho $m = 9$. O TML se manteve em valores baixos, mesmo as séries UCI 3W e UCR *Anomaly Archive* contendo mais observações que aquelas dos conjuntos de dados anteriores. Retornando o foco ao impacto que o uso de lotes deslizantes e abordagem de memória do *Nexus*

tem no custo computacional, é possível observar a diferença de maneira mais explícita por meio da Figura V.13.

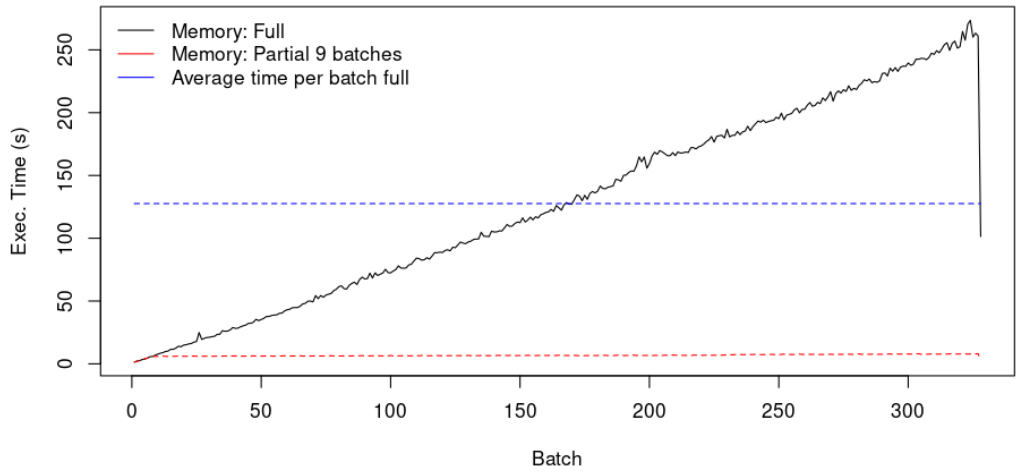


Figura V.13: **Tempo de Execução por Lote (FBIAD) - UCR *Archive*** - Tempo de execução para cada lote da série pH. **Valores de parâmetros**: $w = 243, s = 243$, $m = 0 \rightarrow$ Memória completa e $m = 9 \rightarrow$ Memória parcial.

O gráfico da Figura V.13 apresenta o tempo de execução por lotes para uma das séries do conjunto de dados UCR *Anomaly Archive*. Essa série possui mais de 80.000 observações e a execução do *Nexus* usa mais de 300 lotes para analisá-la em *streaming*. Contudo, o *Nexus* conseguiu processar de maneira adequada a série em *streaming* em relação ao tempo de processamento e impacto de tamanhos de lotes e memória. Percebe-se que a diferença entre a execução com memória completa e parcial cresce significativamente à medida que o *streaming* do *Nexus* percorre a série com uma memória de 9 lotes.

Como se demonstrou ao longo deste capítulo, esse comportamento é um padrão obtido pela conjugação de parâmetros do *Nexus*. Em séries menores, como no exemplo ilustrativo da série pH do conjunto de dados Gecco *Challenge* já se percebe impactos positivos das mudanças de tamanhos de lotes e uso de memória parcial. Nos demais casos, como NAB, Yahoo *Labs* e RARE a diferença de tempo com memória parcial é acentuada e em alguns casos mesmo com memória completa o processamento é acelerado com lotes menores. Isto indica que métodos que mantém resultados equivalentes na acurácia, F_1 e F_1 *softED*, mesmo nas configurações do *Nexus* mais rápidas, são escolhas adequadas quando o custo computacional é muito relevante.

V.3.3 Análise de *Lag*

Para avaliar a capacidade de detecção precoce, além do tempo de execução avaliado na subseção anterior, é importante que os métodos não atrasem para detectar eventos. Para avaliar essa questão

e complementar as análises do exemplo ilustrativo da série pH, esta subseção avalia o *lag* das detecções. Essa avaliação contribui também avaliar o comportamento dos métodos ao longo do *streaming*.

A Tabela V.10 apresenta a análise de *Lag* dos três conjuntos de dados de tecnologia. Para proporcionar uma visão ampliada do comportamento dos métodos nessas séries são listados os valores da mediana de Lag_i^s de cada conjunto de dados. A exploração da tabela permite tanto avaliar o mesmo método para diferentes conjuntos de dados quanto os métodos com menores e maiores atrasos para cada conjunto de dados isoladamente.

Tabela V.10: **Nexus: Análise de *Lag* para conjuntos de dados de Tecnologia** - Comparação da mediana do *lag* para Yahoo *Labs*, NAB e RARE. **Valores de parâmetros**: $w = 243, s = 243, m = 0$. **Valores**: Mediana de todas as séries de cada conjunto de dados.

Método	**Yahoo**	**NAB**	**RARE**
FBIAD	1	9	10
ARIMA	1	11	11
GARCH	1	4	6
CF	1	13	8
LSTM	1	5	10

Para o conjunto de dados YAHOO *Labs* os métodos tem comportamento similar entre si quanto ao *Lag*. A similaridade fica evidenciada pelos valores da Tabela V.10 em que o valor mediano é igual para todos os métodos. A verificação de similaridade de métodos em relação ao Lag_i^s como nas séries do Yahoo *Labs* simplifica a seleção de métodos.

Quando não há diferenças nesse aspecto, a seleção do melhor método pode ser baseada no equilíbrio entre acurácia das detecções e tempo de execução. Porém, pode ocorrer como nas séries do NAB e RARE em que o Lag_i^s é bem distinto entre os métodos. Nesses casos, é necessário avaliar quais métodos conseguem a melhor combinação entre acurácia, tempo de processamento e menores atrasos nas detecções.

Para o NAB observa-se valor elevado para mediana de Lag_i^s que ocorre principalmente em função de algumas séries, cujos *lags* medianos são muito elevados, todos com $Lag_t > 10$. Entretanto, nos dados que originaram o cálculo da mediana no geral para o *lag* ficou próximo de $Lag_t = 3$. Com diferenças tão elevadas entre si como no NAB, reforça-se a importância da avaliação conjunta dos aspectos de acurácia, tempo e atraso mencionado.

O conjunto de dados RARE tem características peculiares como muitas séries sem mudanças ao longo do tempo, mas consideradas com presença de anomalias segundo os rótulos. Também há mudanças muito pontuais em alguns intervalos da série sem que os rótulos de eventos coincidam com os momentos no tempo em que as maiores mudanças na série ocorrem. Assim, há muitos falsos negativos nas detecções e *lags* elevados como se vê na Tabela V.10 em que apenas dois métodos tem

mediana de $Lag_t \leq 10$.

Dentre os três conjuntos de dados de tecnologia, o NAB tem a maior diferença entre valores de Lag_i^s. Como um exercício para demonstração da intuição de análise de equilíbrio entre as métricas disponíveis no *Nexus* nota-se que GARCH e LSTM tem melhores resultados de Lag_i^s. Ao visualizar os resultados consolidados do conjunto de dados NAB no diagrama da Figura V.14 a área de concentração dos dados do método LSTM parece ser ligeiramente maior.

Na Figura V.14 percebe-se também presença de muitos *outliers* nos valores de Lag_i^s. Porém, apenas a análise visual não parece ser conclusiva quanto à diferença de *lag*. Para verificar se a diferença de valores entre GARCH e LSTM tem significância estatística, a Tabela V.11 apresenta o resultado do teste *Wilcox Effect Size*[7] [Kassambara, 2023].

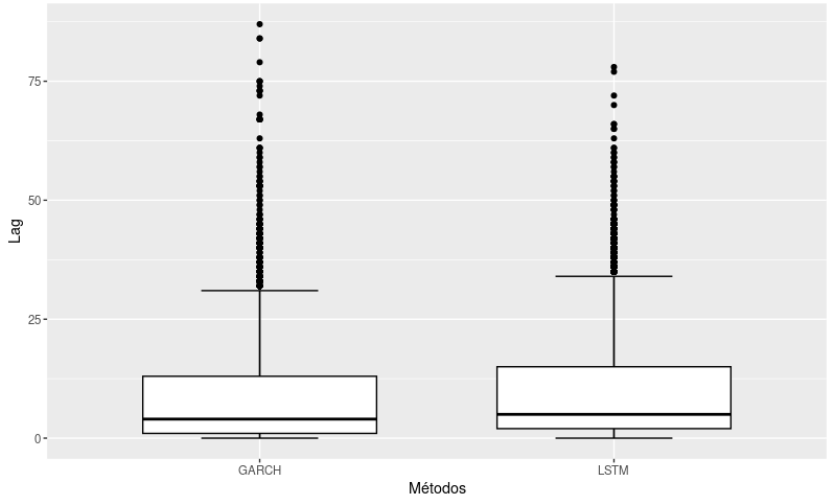


Figura V.14: **Diagrama de caixa Lag_i^s NAB (GARCH e LSTM)** - Métodos com menor *lag* para todas as séries. **Parâmetros**: $w = 243, s = 243, m = 0$

Tabela V.11: **Teste estatístico *Wilcox Effect Size* NAB**: Análise de magnitude de diferença de resultados entre GARCH e LSTM. **Parâmetros**: $w = 243, s = 243, m = 0$

Métrica	**Diferença**	***Effect size***	**Magnitude**
Lag	Sim	0,054	Pequena
TML	Sim	0,859	Grande
Acurácia	Sim	0,728	Grande
F_1 ***SoftED***	Sim	0,583	Grande

A Tabela V.11 confirma que há diferença nos valores de *Lag* com significância estatística entre GARCH e LSTM. Para uma avaliação mais completa, foi realizado o mesmo teste para as métricas TML, acurácia e F_1 *SoftED*. Esse teste compara todos os resultados e gera um valor denominado *effect size*, cuja magnitude do tamanho da diferença é classificado em uma escala como inexistente, pequena, moderada ou grande. Nessa comparação apenas a diferença de Lag_i^s foi classificada como

[7]Disponível no pacote R *rstatix*

pequena, enquanto TML, acurácia e F_1 *SoftED* possuem diferenças classificadas como grandes.

Este capítulo abordou desde as configurações dos experimentos para facilitar sua reprodutibilidade até os resultados obtidos em diversos cenários de uso. O *Nexus* permite a reprodutibilidade por meio da indicação da forma de implementação apresentada, dos métodos e parâmetros usados. Além disso, o *Nexus* pode ser usado em outras condições de configurações para aplicações práticas e pesquisas.

O exemplo ilustrativo com a série *pH* do conjunto de dados *Gecco Challenge* introduz os primeiros resultados do *Nexus*. Com esse caso simplificado, mas aplicado de maneira abrangente, é possível entender a intuição do funcionamento dos parâmetros relacionados ao *warmup*, tamanho de lotes e abordagens de memória. Ademais, como a série é explorada desde as análises gráficas e estatísticas mais básicas até a detecção de eventos em *streaming* fica evidente como o *Nexus* se insere na análise de séries temporais nessa área de pesquisa.

Os diferentes cenários de uso também aprofundam a importância das informações do comportamento dos métodos em *streaming* fornecidas pelos cálculos de Lag_i^s e $DP(x_i)$. Como se demonstra especialmente na exploração das séries do NAB essas informações sobre comportamento das detecções ao longo do *streaming* enriquecem as bases para adequada seleção de métodos. Segundo o melhor conhecimento dos autores, este tipo de detalhe sobre o *streaming* de séries temporais é característica única do *Nexus* e reforça sua relevância.

Não foi explorada otimização dos parâmetros dos próprios métodos para melhorar sua acurácia. Contudo, fica demonstrado que o nível de aprofundamento da análise do comportamento dos métodos em *streaming* proporcionado pelo *Nexus* contribui para o avanço das pesquisas nessa área. Adicionalmente, como o *Nexus* permite a adição de novos métodos, as possibilidades de experimentação e combinação entre parâmetros o habilita como um *framework* de elevada relevância para a área de detecção de eventos em *streaming*.

Por fim, aprofundam-se as possibilidades de uso do *Nexus* em séries com comportamento e tipos de evento distintos, representação de várias áreas de domínio e diferentes complexidades da tarefa de detecção. Uma vez que a Seção V.3 tem essa maior abrangência, seus resultados contribuem para suportar a generalização dos impactos observados nos experimentos iniciais. Em resumo, os achados da pesquisa viabilizados pelo *Nexus* e discutidos neste capítulo corroboram a hipótese que o uso de lotes deslizantes aliado às diferentes abordagens de memória e configurações de tamanho de lote contribuem para detecção de eventos em *streaming*.

Capítulo VI Conclusão

A produção acadêmica sobre detecção *online* apresentou expressivo crescimento nos últimos anos, especialmente na última década. Esse crescimento na pesquisa vai ao encontro da importância do tema nas organizações, marcado pela onipresença da internet e do avanço de dispositivos IoT, plataformas de nuvem e geração de dados em *streaming* em geral. A detecção de eventos *online* é importante nesse contexto porque permite o monitoramento das séries temporais em *streaming* para identificação tempestiva de situações indesejadas. Esses eventos são observações em uma série temporal com significado especial para a área de domínio, tais como identificação de fraudes, prevenção de perdas e acidentes por meio do monitoramento de sensores usados na exploração de petróleo, mudanças inesperadas em séries financeiras, anomalias em dados médicos.

Os trabalhos relacionados analisados cobrem uma diversidade de métodos para detecção de eventos *online*, aplicação de métricas para comparação dos métodos, além de conjuntos de dados e ferramentas para processamento de séries em *streaming*. Contudo, ainda há carência de trabalhos focados especificamente na comparação de métodos de detecção de eventos em dados em *streaming* usando lotes deslizantes de diferentes tamanhos e que sejam independentes de solução específica de processamento de *streaming*. Por isso, este trabalho visou explorar essa lacuna na detecção de eventos *online* em séries temporais em *streaming*, conforme metodologia proposta.

O presente trabalho propôs uma análise do uso de lotes deslizantes na detecção de eventos em séries temporais em *streaming*. Dessa forma, a pesquisa contribuiu para testar diretamente a hipótese relacionada ao uso de lotes deslizantes que consigam lidar com subsequências menores da série poderia resultar em detecção precoce de eventos e redução do custo computacional do processamento *online*, mas sem os impactos negativos da detecção tardia do processamento *offline*.

Além disso, a realização a avaliação experimental abrangente contribuiu para: (i) avaliar o impacto na velocidade da capacidade de resposta aos eventos pela experimentação de diferentes configurações de lotes deslizantes em séries em *streaming* (tamanho de cada lote e memória dos dados); (ii) comparação de diferentes métodos de detecção *online* quanto à capacidade de detectar eventos corretamente e ao tempo de execução.

A metodologia proposta foi implementada na linguagem R por meio do *framework Nexus*. O *Nexus* aplica o uso de lotes deslizantes na detecção de eventos em séries temporais em *streaming*. A arquitetura que embasou a implementação integra detectores de eventos e o gerenciamento da

detecção em *streaming*. O *Nexus* é independente de plataformas, permite tanto adaptar métodos existentes quanto a adição de novos métodos e testar diversas configurações de tamanho de lotes, memória e *warmup*.

Outro aspecto único do *Nexus* é a capacidade de avaliar o comportamento dos métodos ao longo do *streaming* ao invés de apenas avaliar métricas de resultado das detecções. Isto é possível por meio das métricas Lag_i^s e $DP(x_i)$ propostas neste trabalho. Essas métricas auxiliam, respectivamente, na avaliação de atrasos nas detecções e a probabilidade de uma observação da série ser um evento real segundo a frequência que determinado método o detectou ao longo do *streaming*.

Os experimentos indicaram a versatilidade do *Nexus* para aplicar diferentes métodos na detecção de eventos em séries em *streaming*. Essa possibilidade ocorreu mesmo em casos de métodos desenvolvidos inicialmente para detecção *offline*. Por outro lado, como não se conhece previamente a distribuição das séries em *streaming*, uma limitação do *Nexus* é a impossibilidade de escolher antecipadamente o método mais adequado para cada tipo de série.

Na maioria dos experimentos foi observado que o uso de memória parcial dos lotes contribuiu para maior velocidade das execuções. Mesmo com ganho de velocidade, as métricas avaliadas demonstraram que as reduções não foram significativas. A implementação atual do *Nexus* repete as etapas de treinamento ao longo do *streaming*, isto é uma limitação cuja eliminação poderá aumentar a velocidade mesmo no uso de memória completa dos lotes.

Para explorar o comportamento dos métodos avaliados e as diversas configurações de parâmetros do *Nexus* foi realizado um trabalho de seleção e estruturação de conjuntos de dados de séries temporais. A seleção dos conjuntos de dados se deu principalmente pela sua identificação nos trabalhos relacionados relativos à comparação de métodos. Esses conjuntos de dados foram publicados em um repositório público denominado *DAL Events Datasets*.

Na avaliação experimental dos resultados observou-se, já nos experimentos do exemplo ilustrativo, que as configurações de tamanho e memória dos lotes deslizantes impacta a velocidade de capacidade de resposta. Os métodos apresentaram boa velocidade na série pH, devido à pequena quantidade de observações, mesmo nas configurações mais lentas com memória completa e lotes maiores. Ainda assim, métodos como FBIAD, ARIMA e CF conseguiram reduzir o tempo de processamento entre 23.8% e 88% com uso de memória parcial ou lotes menores.

Ressalta-se que as reduções de tempo de processamento, medida pelo TML, de maneira geral não geraram reduções significativas nas métricas de detecção acurácia, F_1 e F_1 *SoftED*. Além disso, como ilustrado nos experimentos, a possibilidade de filtros das detecções ao longo do *streaming* segundo a análise de probabilidade $DP(x_i)$ permite selecionar detecções de maneira mais acuradas. Por exemplo, ao avaliar apenas os eventos detectados com $P(E) > 0.8$, ou seja, marcados em 80% das vezes em que a observação foi analisada ao longo do *streaming* contribuiu para redução de falsos

positivos.

À medida que os experimentos com mais conjuntos de dados com complexidade e tamanhos diferentes são avaliados, demonstrou-se que esse comportamento foi um padrão do *Nexus*. Conforme apontado na avaliação experimental, nos conjuntos de dados NAB, Yahoo *Labs* e RARE o impacto no tempo de processamento sem redução dos níveis de acurácia foram ainda mais relevantes. Esses resultados corroboram com a confirmação da hipótese que o uso de lotes deslizantes aliado à capacidade do *Nexus* de configurações de tamanhos de lotes e abordagens de memória contribuem para detecção precoce de eventos, redução de tempo de processamento sem deteriorar a capacidade de acerto dos métodos.

Além das contribuições diretas deste trabalho, ao longo da pesquisa houve uma série de contribuições para o estudo da detecção de eventos oriundas de resultados preliminares. Os resultados desses trabalhos já foram publicadas em artigos no âmbito do mestrado [Lima et al., 2022a,b; Salles et al., 2023a; Ogasawara et al., 2021; Gea et al., 2021; Escobar et al., 2021]. A Tabela VI.1 sintetiza o histórico das publicações mencionadas.

Tabela VI.1: **Síntese da Produção Acadêmica**: Artigos e artefatos publicados

Trabalho	**Local de Publicação**	**Data**
Escobar et al. [2021]	JIDM	set/2021
Gea et al. [2021]	SBBD	out/2021
Ogasawara et al. [2021]	CILAMCE	nov/2021
Lima et al. [2022a]	IJCNN	jul/2021
Lima et al. [2022b]	DEXEA/SBBD	set/2022
Pacote Harbinger [Salles et al., 2023a]	CRAN R-Project	Jul/2023
Salles et al. [2023b]	*Preprint* a ser submetido	2023
Pacote DAL Events Dataset	A ser submetido ao CRAN R-Project	2024
Pacote Nexus	A ser submetido ao CRAN R-Project	2024

Uma dessas contribuições deu origem a uma taxonomia que explora diversos aspectos da detecção *online* de eventos em séries em *streaming* [Ogasawara et al., 2021]. Os cuidados na transformação de dados, pela remoção de anomalias em séries financeiras e seus efeitos na perda de informações foi outra contribuição relevante [Gea et al., 2021]. Também foram explorados os resultados de diversos métodos para avaliar aspectos relacionados ao viés temporal, atrasos e antecipação nas detecções [Escobar et al., 2021].

Outro resultado foi a publicação de um novo método denominado FBIAD [Lima et al., 2022a] no *International Joint Conference on Neural Networks* (IJCNN). Esse método superou a acurácia e tempo de execução de outros métodos. Os resultados do FBIAD foram promissores mesmo quando ele foi comparado com métodos do estado da arte como MAD, mas os experimentos foram realizados em detecções *offline*.

Com intuito de explorar aspectos relacionados às oportunidades advindas da computação em

nuvem foi proposto o *framework Harbinger Nimbus* (HN), cujo protótipo foi implementado na plataforma *Microsoft Azure* com resultados preliminares promissores [Lima et al., 2022b]. O HN contribui para maior conectividade entre os algoritmos de detecção, sistemas geradores de séries temporais e o consumo dos resultados para tomada de decisão tempestiva. Além disso, HN apresenta contribuições relacionadas à versatilidade de *frameworks* não dependentes de ferramentas específicas e passíveis de proporcionar um ambiente em nuvem para experimentar diferentes métodos de detecção.

Além dos artigos publicados, os artefatos gerados durante a pesquisa ou aperfeiçoados para uso nos experimentos também estão listados na Tabela VI.1. Entre os artefatos foi realizado uma evolução do *framework Harbinger* [Salles et al., 2023a] e sua publicação no *Comprehensive R Archive Network* (CRAN), repositório oficial da linguagem R. O *Nexus* e o repositório de conjuntos de dados para detecção de eventos estão disponíveis em repositórios públicos da plataforma GitHub, mas também serão publicados futuramente no CRAN.

Esses trabalhos anteriores contribuíram para levantamento de questões abordadas ao longo desta pesquisa. Foram, ainda, uma base importante para os experimentos usando a metodologia proposta no cenário específico de detecção *online* de eventos em séries em *streaming*. Também serão úteis como um arcabouço técnico e científico para exploração dos trabalhos futuros abordados a seguir.

Os resultados e experimentos produzidos nesta pequisa, abrem caminhos para trabalhos futuros para experimentação e combinação de métodos de detecção de eventos em dados em *streaming*. Por exemplo, a mudança do método aplicado quando a série muda de comportamento ao longo do tempo para maior capacidade de adaptação das detecções e a combinação de mais de um método para melhoria na acurácia das detecções. Outro aspecto que pode ser estudado em trabalhos futuros para adaptabilidade é a criação de um mecanismo de gerenciamento dos momentos adequados para retreinar os modelos ao longo do *streaming*.

Um aspecto adicional que pode ser explorado em trabalhos futuros é análise de opções de otimizações dos parâmetros do *Nexus*. Os parâmetros foram avaliados em diferentes configurações, contudo ainda há espaço para identificar formas de otimizar o valor de tamanho de *warmup*. A exploração mais aprofundada do *warmup* pode auxiliar a quantidade mínima de observações para que os métodos comece a identificar adequadamente eventos na série.

Como a metodologia proposta neste trabalho não trata da implementação de um método novo, algumas questões observados fogem ao escopo do trabalho. Notadamente, as dificuldades que os métodos experimentados encontraram nos últimos conjuntos de dados UCI 3W e UCR *Anomaly Archive* podem ser exploradas por meio do uso do *Nexus* para testar novos métodos. Também é possível em trabalhos futuros usar o aprendizado sobre a área para desenvolver métodos para detectar eventos raros presentes nessas séries.

Outras oportunidades de trabalhos futuros são aquelas relacionadas à integração do *Nexus* com

plataformas de computação em nuvem. Quanto à computação em nuvem, um caminho é a busca pela integração entre ao HN e o *Nexus*. Para tanto, é preciso o aprofundamento da pesquisa para melhoria e tratamento das limitações atuais, como simplificação ou automação da orquestração das tarefas no HN e processo de seleção dos métodos de detecção segundo o tipo de séries no *Nexus*.

O *Nexus* se diferencia da maioria dos trabalhos relacionados de comparação de métodos estudados nesta pesquisa, por proporcionar um *framework* completo para integração, avaliação e implementação de novos métodos para detecção de eventos em *streaming*. Conforme abordado em detalhes na Tabela III.1, algumas características (*framework* e adição de novos métodos) do *Nexus* são encontradas apenas de maneira isolada por trabalhos como Numenta, Harbinger e MAD, enquanto outras lhe são únicas (uso de lotes, avaliação do processamento do *streaming*). Contudo, alguns desses trabalhos relacionados também indicam possibilidades de pesquisas futuras.

Por fim, algumas dessas oportunidades vindas especificamente da comparação qualitativa com trabalhos relacionados são integração do *Nexus* a uma plataforma de *streaming* robusta como *Kafka*. como tratado nos trabalhos de Rettig et al. [2015] e Belacel et al. [2022a]. Trabalhos como MAD de Ren et al. [2019] tratam da implementação de métodos treinados previamente. As possibilidades de métodos pré-treinados ou análise da possibilidade de abordagem de transferência de conhecimento, por sua vez, podem ser úteis para lidar com casos complexos de eventos raros como descritos por Wu and Keogh [2021].

Referências

Ahmad, S., Lavin, A., Purdy, S., and Agha, Z. Unsupervised real-time anomaly detection for streaming data. *Neurocomputing*, 262:134–147, 2017.

Aminikhanghahi, S. and Cook, D. A survey of methods for time series change point detection. *Knowledge and Information Systemss*, page 339–367, 2017.

Ariyaluran Habeeb, R., Nasaruddin, F., Gani, A., Targio Hashem, I., Ahmed, E., and Imran, M. Real-time big data processing for anomaly detection: A Survey. *International Journal of Information Management*, 45:289–307, 2019.

Atzori, L., Iera, A., and Morabito, G. The Internet of Things: A survey. *Computer Networks*, 54(15):2787–2805, 2010.

Belacel, N., Richard, R., Rangavajjala, D. P., and Adhaduk, R. Online Anomaly Detection for Streaming Data Implemented on Top of Kafka, Scikit-Multiflow and River. In Arai, K., editor, *Proceedings of the Future Technologies Conference (FTC) 2021, Volume 3*, Lecture Notes in Networks and Systems, pages 826–836, Cham. Springer International Publishing, 2022a.

Belacel, N., Richard, R., Rangavajjala, D. P., and Adhaduk, R. Online anomaly detection for streaming data implemented on top of kafka, scikit-multiflow and river. In Springer, editor, *Proceedings of the Future Technologies Conference (FTC) 2021*, pages 826–836, 2022b.

Carmona, R. *Statistical Analysis of Financial Data in R.* Springer Science & Business Media, 2013.

Carter, K. and Streilein, W. Probabilistic reasoning for streaming anomaly detection. In *2012 IEEE Statistical Signal Processing Workshop, SSP 2012*, pages 377–380, 2012.

Chandola, V., Banerjee, A., and Kumar, V. Anomaly detection: A survey. *ACM Computing Surveys*, 41(3), 2009.

Dancho, M. and Vaughan, D. anomalize: Tidy Anomaly Detection. Technical report, https://CRAN.R-project.org/package=anomalize, 2020.

Escobar, L., Salles, R., Lima, J., Gea, C., Baroni, L., Ziviani, A., Pires, P., Delicato, F., Coutinho, R., Assis, L., and Ogasawara, E. Evaluating Temporal Bias in Time Series Event Detection Methods. *Journal of Information and Data Management*, 12(3), 2021.

Esling, P. and Agon, C. Time-series data mining. *ACM Computing Surveys*, 45(1), 2012.

Fu, T.-C. A review on time series data mining. *Engineering Applications of Artificial Intelligence*, 24(1):164–181, 2011.

Gea, C., Lima, J., Bezerra, E., and Ogasawara, E. Análise de métodos de tratamento de outliers para predição dos retornos de índices de ações negociados em bolsa. In *Anais do Simpósio Brasileiro de Banco de Dados (SBBD)*, pages 277–282. SBC, 2021.

Gensler, A. and Sick, B. Performing event detection in time series with SwiftEvent: an algorithm with supervised learning of detection criteria. *Pattern Analysis and Applications*, 21(2):543–562, 2018.

Guralnik, V. and Srivastava, J. Event Detection from Time Series Data. In *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '99, pages 33–42, New York, NY, USA. ACM, 1999.

Han, J., Kamber, M., and Pei, J. *Data Mining: Concepts and Techniques*. Elsevier, 2012.

Hanssens, D. M., Parsons, L. J., and Schultz, R. L. *Market Response Models: Econometric and Time Series Analysis*. Springer, Boston, Mass., 2nd edition edition, 2003.

Hasani, Z. Robust anomaly detection algorithms for real-time big data: Comparison of algorithms. In *2017 6th Mediterranean Conference on Embedded Computing, MECO 2017 - Including ECYPS 2017, Proceedings*, 2017.

Hasani, Z. Anomaly detection algorithms for streaming data: Performance comparison. *Journal of Computer Science*, 16(7):950–955, 2020.

Hiraman, B., Viresh, M., and Abhijeet, C. A Study of Apache Kafka in Big Data Stream Processing. In *2018 International Conference on Information, Communication, Engineering and Technology, ICICET 2018*, 2018.

Iwashita, A. and Papa, J. An Overview on Concept Drift Learning. *IEEE Access*, 7:1532–1547, 2019.

Kassambara, A. rstatix: Pipe-friendly framework for basic statistical tests. Technical report, https://cran.r-project.org/package=rstatix, 2023.

Lima, J., Fernandes, P. A., Salles, R., Escobar, L., Porto, F., Pacitti, E., Coutinho, R., and Ogasawara, E. Forward and backward inertial anomaly detector: A novel time series event detection method. In *Proceedings of IEEE International Joint Conference on Neural Networks 2022 (IJCNN 2022)*. IEEE, 2022a.

Lima, J., Salles, R., Escobar, L., Gea, C., Fernandes, P. A., Pacitti, E., Porto, F., Coutinho, R., and Ogasawara, E. Towards a cloud-based framework for online and integrated event detection. In *Anais Estendidos do XXXVII Simpósio Brasileiro de Bancos de Dados (SBBD)*. SBC, 2022b.

Lin, J., Keogh, E., Lonardi, S., and Chiu, B. A symbolic representation of time series, with implications for streaming algorithms. In *Proceedings of the 8th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery, DMKD '03*, pages 2–11, 2003.

Lomio, F., Baselga, D., Moreschini, S., Huttunen, H., and Taibi, D. RARE: A labeled dataset for cloud-native memory anomalies. In *MaLTeSQuE 2020 - Proceedings of the 4th ACM SIGSOFT International Workshop on Machine-Learning Techniques for Software-Quality Evaluation, Co-located with ESEC/FSE 2020*, pages 19–24, 2020.

Marik, R. and Bohac, L. Non-stationary Events Detection Based on Extrema Value Theory. In *2018 5th International Conference on Systems and Informatics, ICSAI 2018*, pages 889–894, 2019.

Moritz, S., Rehbach, F., Chandrasekaran, S., Rebolledo, M., and Bartz-Beielstein, T. GECCO Industrial Challenge 2018 Dataset: A water quality dataset for the 'Internet of Things: Online Anomaly Detection for Drinking Water Quality'. Technical report, https://zenodo.org/record/3884398, 2018.

Munir, M., Siddiqui, S., Chattha, M., Dengel, A., and Ahmed, S. FuseAD: Unsupervised anomaly detection in streaming sensors data by fusing statistical and deep learning models. *Sensors (Switzerland)*, 19(11), 2019.

Ogasawara, E., Martinez, L., De Oliveira, D., Zimbrão, G., Pappa, G., and Mattoso, M. Adaptive Normalization: A novel data normalization approach for non-stationary time series. In *Proceedings of the International Joint Conference on Neural Networks*, 2010.

Ogasawara, E., Salles, R., Escobar, L., Baroni, L., Lima, J., and Porto, F. Online event detection for sensor data. In *Ibero-Latin American Congress on Computational Methods in Engineering*, Rio de Janeiro, RJ, 2021.

Ren, H., Xu, B., Wang, Y., Yi, C., Huang, C., Kou, X., Xing, T., Yang, M., Tong, J., and Zhang, Q. Time-series anomaly detection service at Microsoft. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 3009–3017, 2019.

Rettig, L., Khayati, M., Cudre-Mauroux, P., and Piorkowski, M. Online anomaly detection over Big Data streams. In *Proceedings - 2015 IEEE International Conference on Big Data, IEEE Big Data 2015*, pages 1113–1122, 2015.

Salles, R., Belloze, K., Porto, F., Gonzalez, P., and Ogasawara, E. Nonstationary time series transformation methods: An experimental review. *Knowledge-Based Systems*, 164:274–291, 2019.

Salles, R., Escobar, L., Baroni, L., Zorrilla, R., Ziviani, A., Kreischer, V., Delicato, F., Pires, P. F., Maia, L., Coutinho, R., Assis, L., and Ogasawara, E. Harbinger: Um framework para integração e análise de métodos de detecção de eventos em séries temporais. In *Anais do Simpósio Brasileiro de Banco de Dados (SBBD)*, pages 73–84. SBC, 2020.

Salles, R., Lima, J., Baroni, L., Castro, A., Carvalho, L., Borges, H., Carvalho, D., Coutinho, R., Bezerra, E., Pacitti, E., Porto, F., Ogasawara, E., and Janeiro (CEFET/RJ), F. C. f. T. E. o. R. d. harbinger: An unified time series event detection framework. Technical report, https://CRAN.R-project.org/package=harbinger, 2023a.

Salles, R., Lima, J., Coutinho, R., Pacitti, E., Masseglia, F., Akbarinia, R., Chen, C., Garibaldi, J., Porto, F., and Ogasawara, E. Softed: Metrics for soft evaluation of time series event detection. Technical report, https://github.com/cefet-rj-dal/softed, 2023b.

Shumway, R. H. and Stoffer, D. S. *Time Series Analysis and Its Applications: With R Examples*. Springer, 2017.

Singh, N. and Olinsky, C. Demystifying Numenta anomaly benchmark. In *Proceedings of the International Joint Conference on Neural Networks*, volume 2017-May, pages 1570–1577, 2017.

Stahmann, P. and Rieger, B. Requirements identification for real-time anomaly detection in Industrie 4.0 machine groups: A structured literature review. In *Proceedings of the Annual Hawaii International Conference on System Sciences*, volume 2020-January, pages 5738–5747, 2021.

Takeuchi, J.-I. and Yamanishi, K. A unifying framework for detecting outliers and change points from time series. *IEEE Transactions on Knowledge and Data Engineering*, 18(4):482–492, 2006.

Talagala, P., Hyndman, R., Smith-Miles, K., Kandanaarachchi, S., and Muñoz, M. Anomaly Detection in Streaming Nonstationary Temporal Data. *Journal of Computational and Graphical Statistics*, 29(1):13–27, 2020.

Truong, C., Oudre, L., and Vayatis, N. Selective review of offline change point detection methods. *Signal Processing*, 167, 2020.

Vyas, S., Tyagi, R., Jain, C., and Sahu, S. Literature review: A comparative study of real time streaming technologies and Apache Kafka. In *Proceedings - 2021 4th International Conference on Computational Intelligence and Communication Technologies, CCICT 2021*, pages 146–153, 2021.

Wu, R. and Keogh, E. Current Time Series Anomaly Detection Benchmarks are Flawed and are Creating the Illusion of Progress. *IEEE Transactions on Knowledge and Data Engineering*, 2021.

Zeileis, A., Leisch, F., Hornik, K., and Kleiber, C. strucchange: An r package for testing for structural change in linear regression models. Technical report, https://cran.r-project.org/package=strucchange, 2002.

Zhang, M., Guo, J., Li, X., and Jin, R. Data-driven anomaly detection approach for time-series streaming data. *Sensors (Switzerland)*, 20(19):1–17, 2020.