UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE INFORMÁTICA
PROGRAMA DE GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

INTERFACE CONVERSACIONAL BASEADA EM LARGE LANGUAGE MODELS (LLMS) PARA DETECÇÃO DE ANOMALIAS SEMÂNTICAS

José Danilo Silva do Carmo

Trabalho de Graduação

Recife

2025

José Danilo Silva do Carmo

INTERFACE CONVERSACIONAL BASEADA EM LARGE LANGUAGE MODELS (LLMS) PARA DETECÇÃO DE ANOMALIAS SEMÂNTICAS

Trabalho apresentado ao Programa de Graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco, como requisito parcial para obtenção do grau de Graduado em Ciência da Computação.

Área de Concentração: Mineração de Processos

Orientador (a): Ricardo Massa Ferreira Lima

Recife

2025

Ficha de identificação da obra elaborada pelo autor,
através do programa de geração automática do SIB/UFPE

Carmo, José Danilo Silva do.
Interface conversacional baseada em Large Language Models (LLMs) para Detecção de Anomalias Semânticas / José Danilo Silva do Carmo. - Recife, 2025.
42 p. : il., tab.

Orientador(a): Ricardo Massa Ferreira Lima
Trabalho de Conclusão de Curso (Graduação) - Universidade Federal de Pernambuco, Centro de Informática, Ciências da Computação - Bacharelado, 2025.
Inclui referências, apêndices.

1. Mineração de Processos. 2. Análise de Conformidade. 3. Detecção de Anomalias. 4. Interface Conversacional. 5. Inteligência Artificial. 6. Large Language Models. I. Lima, Ricardo Massa Ferreira. (Orientação). II. Título.

000 CDD (22.ed.)

JOSÉ DANILO SILVA DO CARMO

INTERFACE CONVERSACIONAL BASEADA EM LANGUAGE MODELS (LLMS) PARA DETECÇÃO DE ANOMALIAS SEMÂNTICAS

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Ciência da Computação da Universidade Federal de Pernambuco, como requisito parcial para obtenção do título de bacharel em Ciência da Computação.

Aprovado em: 05 / 08 / 2025

BANCA EXAMINADORA

_____

Prof. Dr. Ricardo Massa Ferreira Lima (Orientador)

Universidade Federal de Pernambuco

_____

Prof. Dr. Adriano Lorena Inacio de Oliveira (Examinador Interno)

Universidade Federal de Pernambuco

Dedico este trabalho a Deus, cuja presença me acompanhou e sustentou ao longo dos anos de graduação e nos projetos que dela se originaram. Agradeço por ter me livrado de obstáculos das mais diversas maneiras, por ter atuado por meio de tantas pessoas, e por me dar oportunidades mesmo nos contextos mais desafiadores. Sobretudo, sou grato por me conceder forças para realizar, apresentar e concluir este trabalho. Reconheço que todas as minhas conquistas vêm d'Ele, por Ele e para Ele - que toda a glória seja dada a Ele.

AGRADECIMENTOS

Gostaria de expressar minha gratidão à minha mãe, Josiane, que me ofereceu o suporte essencial para que eu pudesse fazer o meu melhor na graduação. Foi ela quem me acordava nas madrugadas, preparava meu café da manhã e orava por mim antes que eu partisse para a universidade. Agradeço também ao meu pai, Eraldo, que sustentou nossa família enquanto eu ainda não podia contribuir, que sempre me incentivou a estudar e concluir a graduação e que, nas vezes em que estava em casa de madrugada, fazia questão de me levar ao ponto de ônibus em seu caminhão. Sem o apoio e o cuidado deles, eu não teria chegado tão longe.

Aos meus primeiros amigos e companheiros de curso, quando eu ainda estava em Sistemas de Informação: vocês me ensinaram, me acolheram e compartilharam comigo momentos inesquecíveis. Em especial, Bruno, Giovani, Alex e Adriel - vocês ajudaram a suavizar as muralhas da universidade. Aos meus últimos amigos de graduação, quando parecia impossível formar bons grupos e criar laços de amizade: Amanda e João Pedro, vocês foram minha motivação para continuar oferecendo o meu melhor em cada disciplina.

Agradeço também ao professor Ricardo Massa, que acreditou em mim mesmo quando eu ainda não havia demonstrado aptidão para a produção científica. Confiar-me a monitoria de sua disciplina, treinar-me na área de mineração de processos e valorizar minhas contribuições nas discussões com os doutores da área foram incentivos fundamentais para que eu seguisse em frente. Não poderia deixar de agradecer a Thiago Araújo por me guiar na execução deste trabalho ao me dar ideias, conselhos e confiança para concluí-lo - sem sua ajuda, eu provavelmente teria desistido mesmo antes de começar.

Ao meu primo Isaías, que me apresentou o Centro de Informática, sem aquela conversa, eu estaria em outra área completamente diferente. Ao próprio Centro de Informática - seu corpo docente, técnicos administrativos e demais funcionários - por ser um espaço de excelência e por tornar tudo isso possível. Ao projeto JuMP, que foi o meio pelo qual aprendi sobre a área de mineração de processos, e a todos os seus integrantes, que me acompanharam ao longo da graduação, vocês me ensinaram a ser o profissional que sou hoje. Por fim, agradeço a todos que, de forma direta ou indireta, fizeram parte dessa fase da minha vida.

RESUMO

Para capturar comportamentos indesejáveis em processos de negócio, são utilizadas técnicas de Análise de Conformidade, que verificam se os processos reais estão alinhados com um determinado modelo de referência - e vice-versa - em busca de desvios e gargalos em relação ao comportamento atual e esperado. Porém, essas técnicas necessitam de um modelo de processo que represente o comportamento esperado, muitas vezes não disponível; isso é mitigado por técnicas de Detecção de Anomalias, que realizam a análise de conformidade diretamente a partir do log de eventos. Uma dessas técnicas é a Explainable Semantic Anomaly Detection (xSemAD) que descobre anomalias numa perspectiva semântica - sem a necessidade de se basear na frequência das atividades - e explica as anomalias através da identificação das restrições violadas utilizando a linguagem Declare. Ainda assim, o resultado dessas técnicas necessita de entendimento de negócio e de mineração de processos, tornando sua utilidade limitada para pessoas do negócio. Para isso, foram utilizadas as Large Language Models (LLMs) que mais recentemente têm se mostrado promissoras em auxiliar na execução e análise de técnicas de mineração de processos. A principal contribuição desta pesquisa reside em utilizar LLMs como interface para o uso da técnica xSemAD em log de eventos, facilitando a utilização da técnica e fornecendo explicações dos desvios para pessoas de negócio de forma contextualizada e em linguagem natural.

Palavras-chaves: Mineração de Processos. Análise de Conformidade. Detecção de Anomalias. Interface Conversacional. Inteligência Artificial. Large Language Models.

ABSTRACT

To capture undesirable behaviors in business processes, Conformance Checking techniques are used to verify whether actual processes align with a given reference model - and vice versa - in search of deviations and bottlenecks between expected and current behaviors. However, these techniques require a process model that represents the expected behavior, which is often unavailable. This is mitigated by Anomaly Detection techniques that perform conformance analysis directly from the event log. One of these techniques is xSemAD, which identifies anomalies from a semantic perspective - without relying on activity frequency - and explains the anomalies by identifying violated constraints using the Declare language. Even so, the results of these techniques require an understanding of business operations and process mining, limiting their usefulness to business professionals. To address this, LLMs, have recently proven promising in supporting the execution and analysis of process mining techniques. The main contribution of this research lies in leveraging LLMs as an interface for using the xSemAD technique on event logs, facilitating the use of the technique and providing contextualized explanations of deviations to business users in natural language.

Keywords: Process Mining. Conformance Checking. Anomaly Detection. Conversational Interface. Artificial Intelligence. Large Language Models.

LISTA DE QUADROS

Quadro 1 – (S1) Role Prompting 27
Quadro 2 – (S3) Abstração Não Técnica, abreviado com "..."para deixar compacto . . 27
Quadro 3 – (S6) Few-Shot Learning, abreviado com "..."para deixar compacto 28
Quadro 4 – (S7) Negative Prompting, abreviado com '...' para deixar compacto 28
Quadro 5 – ReAct Prompt, abreviado com "..."para deixar compacto 29
Quadro 6 – Reposta do algoritmo para 'Diga quais as anomalias que se referem ao início do processo.' 30
Quadro 7 – Resposta do prompt 'Diga quais as anomalias que se referem ao início do processo.' 30
Quadro 8 – Etapas do ReAct (chain-of-thoughts) do modelo 31
Quadro 9 – Resposta do prompt 'Diga quais as anomalias que se referem ao início do processo.' do agente 31
Quadro 10 – Resposta do prompt 'Detecte as anomalias de fim de processo com um limiar de 85%' 32
Quadro 11 – Resposta do prompt 'Detecte as anomalias as maiores anomalias do tipo Succession e Alternate Precedence.' pelo modelo do tipo agente 32
Quadro 12 – Resposta do prompt 'Verifique se existe alguma anomalia de Succession para as atividades 'declaration approved by supervisor' e 'request payment' 33
Quadro 13 – Resposta do prompt 'Detecte as anomalias do tipo Response.' 34

LISTA DE ABREVIATURAS E SIGLAS

LLMs	Large Language Models
NLP	Natural Language Processing
PM4Py	Process Mining for Python
seq2seq	Sequence-to-sequence
TBR	Token Based Replay
TU/e	Eindhoven University of Technology
xSemAD	Explainable Semantic Anomaly Detection

SUMÁRIO

1 INTRODUÇÃO 11
1.1 MOTIVAÇÃO 12
1.2 OBJETIVOS 12
1.2.1 Objetivos Específicos: 12
2 FUNDAMENTAÇÃO TEÓRICA 13
2.1 ANÁLISE DE CONFORMIDADE 14
2.2 DETECÇÃO DE ANOMALIAS 15
2.3 DETECÇÃO DE ANOMALIAS SEMÂNTICAS 16
2.4 XSEMAD: DETECÇÃO DE ANOMALIAS SEMÂNTICAS EXPLICÁVEIS . 18
2.5 LARGE LANGUAGE MODELS 20
2.5.1 LLMs aplicada em Mineração de processos 21
2.5.2 Abordagem escolhida 21
2.5.3 Técnicas de Engenharia de Prompt 22
2.5.4 Agentes 23
3 METODOLOGIA 24
3.1 REVISÃO DA LITERATURA 24
3.2 ANÁLISE DO DATASET 24
3.3 ARQUITETURA E IMPLEMENTAÇÃO 26
4 RESULTADOS 30
5 DISCUSSÃO E TRABALHOS FUTUROS 36
5.1 DISCUSSÃO 36
5.2 TRABALHOS FUTUROS 37
6 CONCLUSÃO 38
REFERÊNCIAS 39
APÊNDICE A – DIRECT ANSWER PROMPT 41
APÊNDICE B – REACT PROMPT 43

1 INTRODUÇÃO

A área de estudo sobre Mineração de Processos busca descobrir, monitorar e aprimorar processos com base em logs de eventos, e é constituída por três conjuntos de funções: Descoberta, Conformidade e Aprimoramento. A Análise de Conformidade, por sua vez, compara os processos reais encontrados no log de eventos com um determinado modelo de referência e vice-versa, em busca de desvios e gargalos.

A característica principal da análise de conformidade tradicional que é avaliar um log de eventos a partir de um modelo de negócio torna-se um problema quando não se tem um modelo de negócio de referência. Além disso, essas técnicas focam apenas em mensurar quão correspondentes estão os dados/modelo analisado com o desejado, sem serem capazes de listar quais são as inconformidades para extrair ações práticas.

Por fim, quando finalmente é realizada uma análise de conformidade, ainda que possua resultados precisos, eles se mostram difíceis de interpretar, por vezes demandando um conhecimento técnico na área de mineração de processos ou carecendo de um contexto específico das regras de negócio para que se possa identificar mais desvios, explicar por que são um problema e apontar o impacto deles para o negócio.

Esses fatores contribuem para que a análise de conformidade tradicional não seja suficiente para ser utilizada por pessoas de domínio do negócio, usuários leigos e não técnicos da área de mineração de processos. Para resolver esses problemas, diversos avanços na área foram feitos, tanto no sentido de aumentar a simplicidade de uso e robustez dos algoritmos de análises de conformidade, quanto no sentido de fornecer uma interface em linguagem natural para realizar tarefas de mineração de processos, democratizando seu uso.

Esta pesquisa propõe-se a facilitar o entendimento e enriquecer a análise de conformidade, fornecendo um artefato que utiliza LLMs como interface para executar as técnicas e explicar seus resultados. Tornando assim os resultados da análise de conformidade mais fáceis de entender e mais ricos para os profissionais das diversas áreas.

A questão central da pesquisa é: "Como a utilização de modelos de LLMs pode enriquecer e facilitar o entendimento da análise de conformidade?". Para fazer isso, serão analisadas técnicas capazes de resolver os problemas supracitados e as melhores abordagens da interação entre LLMs e mineração de processos para facilitar o uso dessas técnicas.

1.1 MOTIVAÇÃO

As técnicas de mineração de processos são ferramentas valiosas pelas quais as organizações podem obter ideias sobre o comportamento real de seus processos de negócio; entretanto, o valor de seus resultados é frequentemente ofuscado pela dificuldade de execução de suas técnicas e pela necessidade de conhecimento técnico e negocial para traduzir seus resultados em ações práticas, especialmente quando se fala de Análise de Conformidade.

Essa pesquisa busca traduzir a terminologia técnica dos desvios identificados pela análise de conformidade em linguagem natural compreensível para os profissionais de diversas áreas e, dessa forma, superar as limitações da análise de conformidade tradicional, representando um avanço de valor científico para a área de mineração de processos ao preencher a lacuna entre a interseção de LLMs e análise de conformidade, fornecendo uma forma de unir técnicas da análise de conformidade e interface em linguagem natural para mineração de processos.

1.2 OBJETIVOS

O objetivo geral deste trabalho é desenvolver um framework computacional que sirva como interface conversacional para executar algoritmos de análise de conformidade por meio de LLMs, facilitando a execução da análise, melhorando o entendimento de seus resultados e possibilitando a extração de ações práticas a partir dessas técnicas.

1.2.1 Objetivos Específicos:

Para alcançar o objetivo geral, os seguintes objetivos específicos serão perseguidos:

- **OE1:** Realizar uma revisão da literatura sobre os temas: Mineração de processos, com ênfase em análise de conformidade, LLMs focando em compreensão de linguagem natural e geração de explicações, por fim trabalhos que relacionam LLMs e mineração de processos para analisar as melhores práticas;
- **OE2:** Escolher métodos de análise de conformidade que possam fornecer insights, testá-los e prepará-los para serem utilizados no framework;
- **OE3:** Implementar o framework computacional que por meio de linguagem natural realiza as análise de conformidade escolhidas e gera explicações em linguagem natural;

2 FUNDAMENTAÇÃO TEÓRICA

A área de mineração de processos é uma disciplina situada na interseção entre ciência de dados e gerenciamento de processos de negócio. Seu objetivo principal é descobrir, monitorar e aprimorar processos de negócio por meio da extração de conhecimento dos logs de eventos disponíveis em sistemas de informações. Esses logs de eventos são como um diário digital, detalhando quais atividades (eventos) foram executadas, quando, por quem e para qual caso.

A estrutura mínima essencial desses logs contém um identificador único do caso (podendo ser chamado de *case:concept:name*), o nome ou descrição da atividade processual executada (também chamada de evento, atividade ou *concept:name*), e a identificação de data/hora (*timestamp*) da sua ocorrência. Por fim, para essa pesquisa é importante definir que um conjunto ordenado de atividades será chamado de "trilha", ou seja, a sequência de eventos, podendo representar toda a instância de um caso ou uma parte dele.

Tabela 1 – Exemplo de Log de Eventos

	Case	Activity	Timestamp	Resource
0	76457	start trip	2016-10-05 00:00:00	EMPLOYEE
1	76457	end trip	2016-10-05 00:00:00	EMPLOYEE
2	76457	permit submitted by employee	2017-04-06 13:32:10	EMPLOYEE
3	76457	permit final approved by supervisor	2017-04-06 13:32:28	SUPERVISOR
4	76457	declaration submitted by employee	2017-04-07 13:38:14	EMPLOYEE
5	76457	declaration final approved by supervisor	2017-04-07 13:40:17	SUPERVISOR
6	76457	request payment	2017-04-11 15:03:43	UNDEFINED
7	76457	payment handled	2017-04-13 17:30:53	UNDEFINED
8	76667	start trip	2016-11-21 00:00:00	EMPLOYEE
n	...	...	...	...

Fonte: Elaborada pelo autor (2025)

Os logs de eventos podem conduzir três tipos de mineração: A primeira sendo a Descoberta, que produz um modelo de processo (e.g., um diagrama BPMN ou rede de Petri) a partir dos logs de eventos, a segunda sendo a Análise de Conformidade, que compara um modelo existente com o comportamento real, e por fim o Aprimoramento de processos, que tem o objetivo de redesenhar, otimizar e melhorar o modelo e os processos de negócio a partir dos insights das etapas anteriores (AALST, 2012).

2.1 ANÁLISE DE CONFORMIDADE

Esta pesquisa se concentra na Análise de Conformidade (conformance checking) que compara o comportamento do modelo *"as-is"* registrado nos logs de eventos e um modelo *"to-be"* que serve como referência de como os processos deveriam ser executados, buscando encontrar, quantificar e analisar os desvios entre eles. As técnicas mais tradicionais são:

- **Algoritmos de Repetição do Log (Token Based Replay (TBR)):** Cria um token para cada caso e executa evento por evento, no modelo, movendo o token pelo modelo e verificando se o movimento é válido (se existe uma transição para este movimento). No fim, são verificados os tokens produzidos, consumidos, faltantes (tokens artificiais criados para suprir movimentos inválidos) e remanescentes (que não chegaram ao fim) para se contabilizar o resultado.
- **Algoritmos de Alinhamento de trilhas (Trace Alignment):** Busca encontrar o alinhamento (sequência de pares) ótimo entre um trilha do log e a sequência de execução mais próxima permitida pelo modelo por meio de uma função de custo, conseguindo expressar os desvios de forma mais granular.
- **Algoritmos Baseados em Restrições (Constraint-based):** Diferente dos demais, esse não especifica um caminho rígido mas sim um conjunto de regras que devem ser respeitadas, ou seja, tudo é permitido a menos que seja explicitamente proibido, assim cada trilha do log de eventos é avaliado por essas regras e restrições.

De forma geral, todas essas técnicas apenas mensuram a qualidade do modelo observado, com base na comparação com o log de eventos ou vice-versa. Essa qualidade é definida em quatro dimensões (DUNZER et al., 2019), onde cada uma delas é uma porcentagem:

- **Fitness (Adequação):** Representa a capacidade do modelo reproduzir o comportamento observado no log de eventos.
- **Precision (Precisão):** Indica se o modelo evita comportamentos que nunca ocorreram na realidade.
- **Generalization (Generalização):** Capacidade de aceitar comportamentos similares.
- **Simplicity (Simplicidade):** Quantifica quão simples é o modelo.

2.2 DETECÇÃO DE ANOMALIAS

Ao tentar realizar a Análise de Conformidade, o usuário se depara com uma grande limitação: a necessidade de um modelo de processo de referência - um modelo que represente o fluxo de trabalho correto - para ser comparado com o log de eventos real. No entanto, em diversos cenários, esse modelo de referência não existe, exigindo do usuário um grande esforço a priori para criar esse artefato de referência, uma vez que o resultado dessa análise está intrinsecamente ligado à qualidade do modelo de referência.

Além disso, outros problemas podem surgir quando se está analisando um domínio onde as instâncias de processos mudam constantemente, no qual o modelo de referência fica obsoleto rapidamente, exigindo ainda mais esforço dos especialistas para ajustar o modelo, ou seja, é necessário um modelo que seja formal, preciso e abrangente.

Por fim, ainda que o modelo de referência exista, os algoritmos de análise de conformidade tradicionais apenas focam em medir a correspondência do modelo real com o esperado, mas falham em descobrir erros quando representam um comportamento não previsto no modelo.

Para resolver esses problemas, foram desenvolvidos algoritmos de análise de conformidade que não apenas encontram falhas não previstas no log de eventos, como não necessitam de um modelo de referência como base. Essas são as técnicas de Detecção de Anomalias, que tradicionalmente seguem as seguintes intuições (BEZERRA; WAINER; AALST, 2009):

- Um conjunto de execuções de processos pode ser dividido em normais e anômalas.
- Cada execução anômala é "pouco frequente" em comparação às demais execuções.
- Modelos de processo que "explicam" as execuções normais "fazem sentido".
- Modelos que poderiam explicar tanto as execuções normais quanto algumas anômalas "fazem menos sentido".

Nesse caso, esses algoritmos detectam com base na conformidade (quão bem uma trilha se alinha ao modelo) e na infrequência da trilha. Isso pode ser visto na descrição do algoritmo baseado em amostragem proposto por Bezerra e Wainer em (BEZERRA; WAINER; AALST, 2009):

1. **Identificação de Candidatos:** Identifica todos as "trilhas" infrequentes no log, ou seja, as que estão abaixo de um certo limiar, como 2% ou 5% do total.

2. **Descoberta de um modelo:** Seleciona uma amostra aleatória do log de processo, considerando que essa amostra não irá conter as anomalias - uma vez que são raras - e descobre um modelo de processo a partir dessa amostra.

3. **Verificação e Classificação:** Analisa cada trilha candidata, considerando que se ela não for uma instância válida do modelo minerado então é classificado como uma anomalia.

Por meio desse algoritmo, não apenas é possível resolver os problemas supracitados, mas fornece uma abordagem automatizada eficiente sem a necessidade de análise humana prévia e ainda consegue encontrar anomalias que são muito semelhantes às trilhas comuns.

2.3 DETECÇÃO DE ANOMALIAS SEMÂNTICAS

Como explicado anteriormente, os algoritmos tradicionais de detecção de anomalia são baseados em frequência 2.2, onde o conceito de anomalia se refere ao comportamento incomum. Porém, essa característica principal desses algoritmos faz com que os desvios mais relevantes - que ocorrem com certa frequência - não sejam detectados, uma vez que esses algoritmos não observam o significado dos eventos registrados.

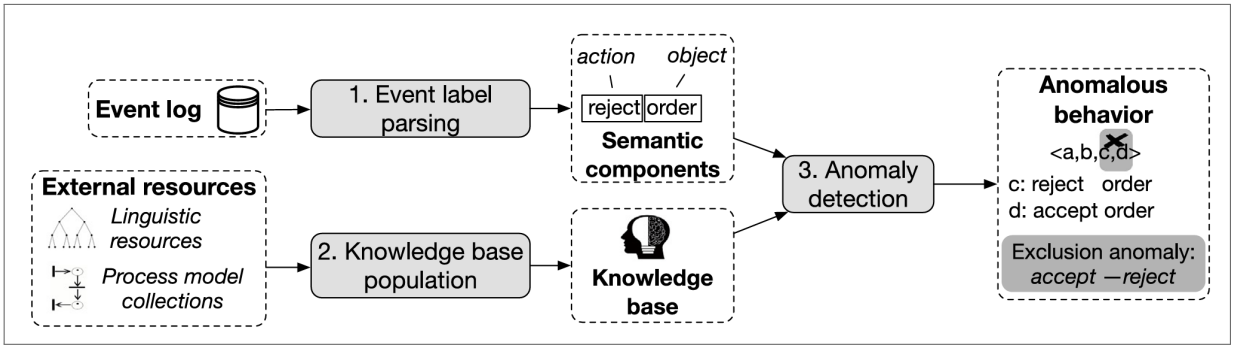
Para entender isso, é possível pensar no caso de um log de eventos onde se tornou normal que o pagamento ocorra antes da aprovação do mesmo ou quando um mesmo pedido é tanto aceito como rejeitado. Nesses casos, uma vez que esses erros se tornaram frequentes, então não serão capturados pela detecção de anomalias baseadas em frequência.

Para preencher essa lacuna, foi desenvolvido um novo algoritmo que explora a linguagem natural associada aos rótulos dos eventos nos logs para a detecção de anomalias. Que é a Detecção de Anomalias Semânticas (van der Aa; REBMANN; LEOPOLD, 2021) para ser possível identificar padrões de execução semanticamente inconsistentes, como os citados acima.

A figura 1 exibe o fluxograma do algoritmo, que é definido pelos três passos a seguir:

1. **Extração de informações semânticas:** O algoritmo utiliza técnicas de Natural Language Processing (NLP), como o modelo BERT, para analisar e extrair objetos de negócio e ações a partir dos rótulos textuais dos eventos. Exemplo: "Reject Order" é classificado como "Reject" (ACT ou ação), que representa o que foi feito e "Order" (BO ou objeto de negócio) que representa em que feito.

Figura 1 – Visão geral da abordagem de detecção de anomalias semânticas



Fonte: van der Aa; REBMANN; LEOPOLD (2021)

2. **Base de conhecimento:** Os padrões extraídos são comparados com uma base de conhecimento que contém regras semânticas gerais sobre como os processos devem ser executados. Essas regras são criadas a partir de recursos (van der Aa; REBMANN; LEOPOLD, 2021) que possibilitam o uso para diversos tipos de processos, com o objetivo de verifica as ações aplicadas à objetos de negócio como o caso onde um objeto de negócio não pode ser "arquivado" antes de "criado", resultando nas relações de:

   - **Ordem:** Se uma ação deve ocorrer antes de outra ("avaliar"antes de "decidir").
   - **Co-ocorrência:** Se uma ação implica a ocorrência de outra ("fabricar" pode resultar em "vender").
   - **Exclusão:** Se duas ações não devem co-ocorrer ("aceitar" e "rejeitar" são antônimos por isso não devem ocorrer juntas para o mesmo objeto de negócio).

3. **Detecção de anomalias semânticas:** Por fim, o algoritmo compara as informações semânticas com a base de conhecimento por meio de igualdade, sinonímia ou similaridade semântica para identificar as seguintes violações:

   - **Violações de Ordem:** Quando eventos ocorrem na sequência oposta à esperada (e.g., um pedido é aprovado antes de ser avaliada).
   - **Violações de Co-ocorrência:** Quando um evento que deveria co-ocorrer com outro está ausente (e.g., uma venda foi feita, mas o produto nunca foi fabricado).
   - **Violações de Exclusão:** Quando eventos que deveriam ser mutuamente exclusivos co-ocorrem para o mesmo objeto de negócio (e.g., um pedido é aceito e rejeitado na mesma instância).

Por fim, o algoritmo também possui um mecanismo para tratar repetições, dividindo as "trilhas" de execução para evitar a detecção de falsos positivos em casos de retrabalho. O resultado do algoritmo é uma coleção de violações detectadas, onde cada uma exibe os eventos associados, a regra que foi violada e o número de ocorrências da violação.

A análise semântica não substitui as abordagens baseadas em frequência; pelo contrário, essa abordagem se torna complementar, uma vez que, ao aplicar a análise semântica e depois refinar com a análise baseada em frequência, é possível alcançar uma detecção de anomalia mais robusta ao detectar tanto as anomalias comuns quanto as incomuns.

2.4 XSEMAD: DETECÇÃO DE ANOMALIAS SEMÂNTICAS EXPLICÁVEIS

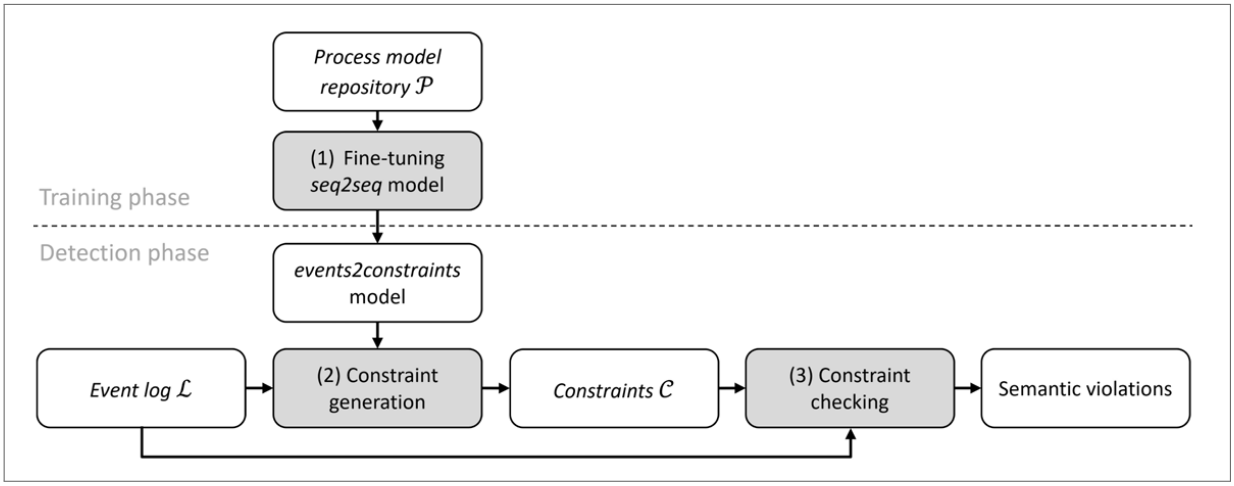
Por meio das abordagens anteriores, foi possível eliminar a necessidade de um modelo de referência 2.2, simplificando a realização de uma análise de conformidade para identificar anomalias frequentes 2.3. Porém, a técnica anterior apenas captura anomalias em pares de eventos que compartilham o mesmo objeto de negócio (BO). Exemplo: "aprovar solicitação" e "rejeitar solicitação" são identificados, pois ambos atuam sobre o objeto "solicitação".

Esse problema foi solucionado pelo trabalho do Caspary et al. (CASPARY; REBMANN; AA, 2023), porém seu resultado apenas identifica os eventos que são anômalos sem identificar qual restrição foi violada, perdendo explicabilidade no processo. Ou seja, é necessária uma abordagem que supere a limitação semântica do trabalho de Van der Aa et al. 2.3, mas que se mantenha acionável ao explicar a natureza das anomalias nos seguintes níveis:

1. **Trace Level:** A "trilha"inteira é marcado como anômala.
2. **Event Level:** Um evento específico ou par de eventos é marcado como anômalo.
3. **Constraint Level:** Exibe a restrição violada pelo (par de) evento(s) violado(s).

Por isso, foi pensada uma abordagem chamada xSemAD (Explainable Semantic Anomaly Detection) (BUSCH; KAMPIK; LEOPOLD, 2024) que utiliza um modelo Sequence-to-sequence (seq2seq) (especificamente o *Flan-T5*) para identificar e explicar anomalias semânticas. Esse algoritmo não apenas identifica as anomalias a nível de "trilha"(1) mas exibe quais os eventos (2) que quebraram a restrição e qual das restrições (3) foi quebrada, fornecendo assim uma base para avaliação dos resultados e busca do problema raiz.

Figura 2 – Visão geral da abordagem de detecção de anomalias semânticas explicáveis



Fonte: BUSCH; KAMPIK; LEOPOLD (2024)

O algoritmo consiste em duas fases (treinamento e detecção) e três componentes centrais (execução do modelo ajustado, geração de restrições e verificação de restrições) como mostrado na figura 2 e detalhado abaixo:

1. **Fase de Treinamento:** Por meio de um repositório de modelos de processo é feito o treinamento que relaciona as relações de execução e determinadas regras. Foram escolhidas regras do modelo DECLARE (CICCIO; MONTALI, 2022) para o treinamento, mas poderia fazer o *fine-tuning* para relacionar com outros padrões. O modelo resultante é chamado de *events2constraints*, para ser utilizado na fase de detecção.

2. **Fase de Detecção de Anomalias:** Diferente da fase de treinamento que só precisa ocorrer uma vez, essa é a fase de execução do algoritmo que utiliza o modelo treinado.

    - **Geração de Restrições:** O modelo *events2constraints* é usado para determinar as restrições do log de eventos e as filtra por meio do algoritmo *beam search*, alguns exemplos de restrições são:

        – Init(a): O processo começa com a atividade "a".

        – End(a): O processo termina com a atividade "a".

        – CoEx(aj,ak): Ambas as atividades aj e ak devem estar no processo.

        – Ch(aj,ak): Ou a atividade aj ou atividade ak devem estar no processo.

        – Resp(aj,ak): Se a atividade aj ocorre, então a atividade ak ocorre depois de aj.

- **Verificação de Restrições:** Cada uma das "trilhas" são verificadas em relação às restrições geradas, para verificar se são anomalias. O resultado é um conjunto de violações semânticas, com a explicação da restrição violada.

O resultado desse algoritmo é uma lista de restrições, identificando qual a restrição foi quebrada, em quais eventos e quantas vezes essa restrição foi violada. Por exemplo:

Alternate Response (declaration approved by supervisor, end trip) 9

Significa que ocorreram **9** violações à restrição ***Alternate Response*** pelo par de atividades **"approved by supervisor"** e **"end trip"**. Dessa forma, é possível fazer uma análise para mitigar esse erro; nesse caso, verificar o que poderia estar causando viagens sem aprovação do supervisor, representando um avanço significativo no entendimento das anomalias.

Dessa forma, não apenas é possível realizar uma análise de conformidade sem a necessidade da elaboração de um modelo de referência prévio, mas essa análise captura as violações mais frequentes, indica quais as restrições foram violadas e em quais eventos elas ocorreram, tornando a análise de conformidade explicável e acionável para os usuários.

2.5 LARGE LANGUAGE MODELS

Apesar desses avanços nas técnicas de análise de conformidade, existem problemas para entender, analisar e interpretar os resultados da mineração de processos, uma vez que é necessário ter conhecimento do negócio e um conjunto de habilidades para transformar os insights coletados em ações práticas. Conforme apontado por Barbieri et al. (tradução livre):

> "O uso dessas ferramentas, no entanto, requer conhecimento da própria tecnologia e é feito principalmente por equipes técnicas (analistas de processos, cientistas de dados e afins)"(BARBIERI et al., 2023).

A técnica xSemAD explicada anteriormente 2.4 é um exemplo disso, quando a explicação da regra violada incide novamente para um conceito técnico que é o caso do DECLARE (CICCIO; MONTALI, 2022), sendo necessário primeiro entender sobre esse conceito de mineração de processos para conseguir extrair insights e tomar decisões.

Esses fatores representam um desafio significativo para usuários não técnicos, limitando o impacto da mineração de processos e, especificamente, da análise de conformidade em aplicações do mundo real (fora do âmbito acadêmico).

LLMs se referem a modelos de linguagem neural baseados em uma arquitetura de *Transformer*, que são pré-treinados em grandes volumes de dados textuais (MINAEE et al., 2025) por meio de um mecanismo de atenção com o objetivo de "aprender"padrões linguísticos para prever o próximo "token" em uma sequência, dado o contexto das palavras anteriores, lhes conferindo a capacidade de compreender, gerar e interagir com a linguagem humana.

Com o advento das LLMs e sua notável capacidade de processar e gerar em linguagem humana, diversas pesquisas na área de mineração de processos têm explorado maneiras de utilizá-las para responder perguntas de mineração de processos, facilitar o uso de suas técnicas ou contextualizar seus resultados por meio de uma interface em linguagem natural, como explorado nos estudos de Berti et al. (BERTI; SCHUSTER; AALST, 2024).

2.5.1 LLMs aplicada em Mineração de processos

O uso de LLMs no contexto de mineração de processos é uma área de pesquisa relativamente recente que se demonstrou promissora ao explorar aspectos como, por exemplo: Identificação de gargalos, comportamentos indesejados (BERTI; SCHUSTER; AALST, 2024) e transformação de descrições processuais em modelos de processos (ZICHE; APRUZZESE, 2024).

Mais recentemente, as LLMs foram utilizadas como *parser semântico* para processar perguntas em operações de mineração de processos em SQL (BARBIERI et al., 2025) e até mesmo como forma de detectar anomalias (REBMANN et al., 2025) por meio de *fine-tuning* de modelos de linguagem para classificação binária de "trilhas"anômalas.

Esses fatores demonstram que LLMs podem auxiliar na compreensão e geração de explicações em linguagem natural dos desvios encontrados, indicando quais regras foram violadas e seu impacto, e traduzir o jargão técnico de mineração de processos para a linguagem natural.

2.5.2 Abordagem escolhida

Vale a pena ressaltar que a proposta de utilizar LLMs para detectar anomalias do trabalho de Rebmann et al. (REBMANN et al., 2025) não resolve o problema aqui descrito, uma vez que apenas avalia se um comportamento de processo é semanticamente coerente ou anômalo, sem explicar o porquê, ou seja, é uma abordagem limitada ao primeiro nível 2.4 de detecção de anomalias, com resultados à mercê das alucinações das LLMs e sem a explicabilidade necessária para tomar decisões com base nos resultados.

Por outro lado, a abordagem da Barbieri et al. (BARBIERI et al., 2025) dá luz para combinar a flexibilidade de uma interface em linguagem natural dos LLMs com o poder, precisão e a escalabilidade das ferramentas existentes, uma vez que os LLMs atuam apenas como *analisador semântico*, ou seja, em vez de usar os LLMs para calcular a resposta, a arquitetura proposta os utiliza para traduzir a pergunta do usuário para uma representação que possa ser executada por ferramentas que podem realizar o cálculo, garantindo a precisão dos resultados.

Ainda assim, esta pesquisa não utilizará completamente a proposta da Barbieri et al. (BARBIERI et al., 2025) pois ela não utiliza diretamente os algoritmos especializados em mineração de processos e propõe uma abordagem híbrida de tradução. O primeiro problema é dar uma responsabilidade parcial para o modelo chegar à resposta, uma vez que a arquitetura utiliza a biblioteca PM4Py (BERTI; ZELST; AALST, 2019) apenas para executar filtragens no log de eventos, de forma a ficar sujeita a respostas erradas.

Outra questão que não será utilizada é a abordagem híbrida de tradução onde a pergunta é primeiramente avaliada por um *"parser"* baseado em regras (mais barato, rápido e previsível) e, caso este não consiga traduzir a pergunta, o *parser* baseado em LLMs é acionado. A abordagem baseada apenas em LLMs se demonstrou superior em precisão e capacidade de generalização, ou seja, se demonstrou capaz de responder a mais perguntas com uma resposta totalmente precisa, mesmo quando são perguntas mais abstratas.

2.5.3 Técnicas de Engenharia de Prompt

A forma pela qual LLMs são utilizados é por meio de instruções (prompts) que são a entrada de texto enviada pelo usuário. O próprio mecanismo de atenção das LLMs faz com que os prompts afetem significativamente o resultado da resposta. Por isso, é necessário fazer engenharia de prompt, que são técnicas para guiar os LLMs em direção aos resultados esperados. O trabalho de Humam Kourani et al. (KOURANI et al., 2024) avaliou algumas estratégias de prompt para mineração de processos:

- **Role Prompting:** Essa estratégia visa evitar respostas genéricas ao declarar um papel para os LLMs como especialista de domínio.
  - **(S1) Especialista em modelagem de processos:** O LLM é instruído a assumir o papel de um especialista em modelagem de processos de negócios e no padrão BPMN 2.0.

  - **(S2) Proprietário do Processo:** O LLM é instruído a assumir o papel de um especialista em processos familiarizado com o domínio do processo fornecido, utilizando seu conhecimento de domínio para preencher quaisquer lacunas ausentes ao analisar o processo.

- **(S3) Abstração Não Técnica:** Instrui o LLM a evitar termos técnicos e a usar linguagem natural para descrever o comportamento do processo subjacente, para tornar as explicações acessíveis não técnicos.
- **(S4) *Chain of Thoughts*:** Solicita que o LLM deve explicar as motivações e as etapas de raciocínio intermediárias para responder à pergunta, para mitigar as alucinações.
- **(S5) *Knowledge Injection*:** Fornece informações específicas de contexto, para ter um resultado análogo ao de *fine-tuning* porém computacionalmente mais barato.
- **(S6) *Few-Shot Learning*:** Fornece alguns pares de entrada de exemplo e saídas esperadas para que o LLM possa aprender com os exemplos.
- **(S7) *Negative Prompting*:** Exemplos que declaram o que ser evitado nas respostas.

Os resultados foram que as estratégias **S1** e **S2** demonstraram impacto mínimo, o **S4** teve efeitos controversos ao gerar respostas mais longas, o **S5** não surtiu tantos efeitos, enquanto a estratégia **S3** se destacou por seus resultados, e a **S6** e **S7** tiveram efeitos positivos.

2.5.4 Agentes

Uma outra maneira de melhorar o resultado dos LLMs é o uso de agentes, que em termos gerais são entidades inteligentes capazes de realizar tarefas específicas no meio do seguinte processo: (1) Perceber o ambiente. (2) Planejar, que é uma das capacidades mais críticas para ser considerado um agente e (3) executar ações (HUANG et al., 2024).

Os agentes de LLM podem seguir vários paradigmas e um deles é o *ReAct* (YAO et al., 2023) que oferece uma estrutura robusta e flexível para construir agentes inteligentes e resolver problemas complexos, no qual, em vez de apenas devolver a resposta final, o LLM irá executar um ciclo de Pensamento, Ação e Observação, no qual a lógica principal é que ele raciocina para agir e age para raciocinar, uma vez que primeiramente cria um plano de ação para determinar o que deve ser feito e, durante a ação, é possível buscar novas informações para pensar.

3 METODOLOGIA

A metodologia adotada para este estudo é definida em três passos com o objetivo de construir e avaliar um artefato computacional: o de analisar os trabalhos correlatos já existentes, implementar o artefato com base nas técnicas mais avançadas já aplicadas na área de mineração de processos e, por fim, avaliar se os resultados cumpriram os objetivos.

3.1 REVISÃO DA LITERATURA

Para fundamentar teoricamente o estudo, concluir o objetivo **OE1** 1.2.1 e garantir que o framework está sendo feito a partir dos estudos mais recentes, foi realizada a seguinte revisão da literatura, para buscar artigos sobre:

- A interseção de LLMs e mineração de processos com o objetivo de conhecer quais são as melhores técnicas utilizadas para criar uma interface em linguagem natural e replicá-las.
- Técnicas de análise de conformidade que possam fornecer insights para as LLMs.
- Benchmarks voltados para o uso de LLMs em mineração de processos, para avaliar abordagens de como avaliar a contribuição científica do trabalho.
- Log de eventos que possam servir de teste para o framework: 3.2

3.2 ANÁLISE DO DATASET

Uma vez que o objetivo é avançar na área a partir do que foi feito, então foi escolhido o mesmo dataset que o artigo (BUSCH; KAMPIK; LEOPOLD, 2024) utilizou para avaliar a efetividade da técnica xSemAD no mundo real, pois este trabalho está voltado para esse mesmo fim. O dataset é o International Declarations Log (KLEIN et al., 2020) do desafio BPI 2020 [1].

Esse dataset possui dados reais do processo de reembolso de despesas de viagens de funcionários da Eindhoven University of Technology (TU/e), possuindo eventos de 2017 (dados de apenas dois departamentos) e 2018 (toda a universidade). Esse log de eventos possui 6.449 casos e 72.151 eventos com os seguintes campos principais:

[1] Dataset disponível em: <https://data.4tu.nl/collections/BPI_Challenge_2020/5065541/1>

Tabela 2 – Descrição dos campos do dataset

Campo	Quantidade Únicos	Tipo
id	69.073	ID
org:resource	2	categorical
concept:name	34	categorical
time:timestamp	51.270	datetime
org:role	8	categorical

Fonte: Elaborada pelo autor (2025)

Além destes campos, existem quatro colunas de atributos para vincular a declaração à sua autorização de viagem, porém há inconsistências entre elas. Também há cinco atributos numéricos, incluindo Amount, RequestedAmount e OriginalAmount, que são quase sempre idênticos. E seis atributos categóricos codificados como IDs, que incluem informações sobre tarefas (6 tipos), orçamentos (207 tipos), projetos (825 tipos) e unidades organizacionais (27 tipos). De acordo com a descrição do dataset, o fluxo é:

1. **Solicitação:** Viagens internacionais precisam pedir permissão para o supervisor.
2. **Submissão:** O processo inicia com a submissão do documento pelo funcionário.
3. **Aprovação Inicial:** A solicitação é enviada para aprovação da administração de viagens.
4. **Aprovação Financeira e Gerencial:** Se aprovada, a solicitação é encaminhada ao responsável pelo orçamento e, em seguida, ao supervisor.
5. **Exceção:** Se o responsável pelo orçamento e o supervisor forem a mesma pessoa, apenas uma dessas etapas é realizada.
6. **Aprovação:** Em alguns casos, o diretor também precisa aprovar a solicitação.

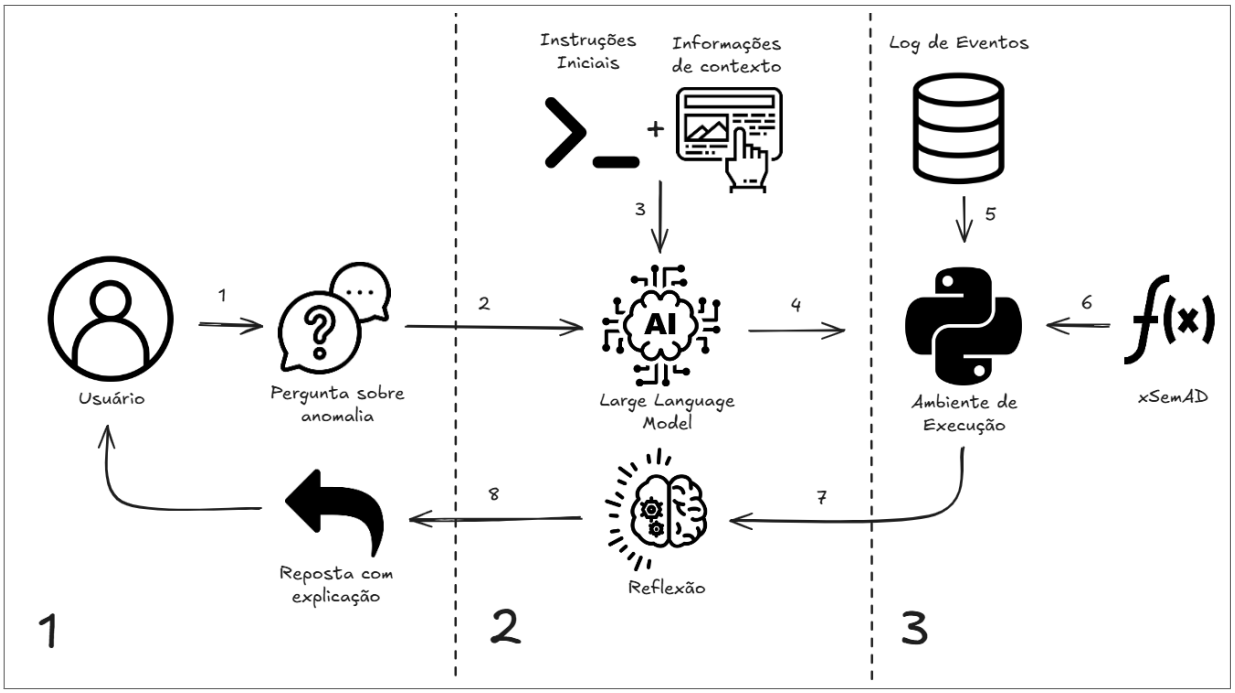
O processo termina com a viagem ocorrendo ou com o pagamento sendo solicitado e efetuado. Caso queira obter o reembolso dos custos de uma viagem, uma declaração é submetida. Isso pode ser feito de duas maneiras:

- Assim que os custos são efetivamente pagos (por exemplo, voos ou taxas de inscrição em conferências).
- Até dois meses após a viagem (para custos como hotel e alimentação, que geralmente são pagos no local).

3.3 ARQUITETURA E IMPLEMENTAÇÃO

A figura 3 exibe de forma geral a arquitetura do projeto, que é composta por 3 fatores: (1) Interação com o Usuário, (2) Interface de LLM e (3) Recursos. No qual o usuário faz as perguntas para a interface LLM, que, munida de instruções e informações quanto ao contexto dos dados, itera ações no ambiente de execução previamente fornecido.

Figura 3 – Visão geral da arquitetura



Fonte: Elaborada pelo autor (2025)

A primeira etapa da arquitetura é conseguir receber perguntas do usuário em linguagem natural e, por conseguinte, receber uma resposta de igual forma, em linguagem natural. O modelo de linguagem escolhido foi o *Gemini 2.5 Pro*, que é um modelo comercial multimodal desenvolvido pelo Google DeepMind e lançado em 17 de junho de 2025 [2].

A escolha desse modelo foi devido à sua janela de contexto de 1 milhão de *tokens* - que dá a possibilidade do modelo receber muitas informações textuais - e à capacidade nativa de fazer *chain-of-thought*, ou seja, raciocinar através de etapas antes de formular uma resposta, que é uma etapa crucial para a implementação de agentes inteligentes 2.5.4.

Para fornecer a capacidade de raciocínio para o modelo de linguagem, foi escolhido implementar o paradigma de agentes conhecido como *ReAct* 2.5.4, dessa forma, o modelo irá "pensar" antes de cada ação tomada e assim chegar a uma explicação detalhada dos resultados.

[2] <https://blog.google/technology/google-deepmind/gemini-model-thinking-updates-march-2025/>

Dado que nem todos os modelos de linguagem possuem a capacidade de se tornarem agentes, então também foi implementada uma versão baseada apenas em resposta direta que implementa todas as estratégias de prompt que serão explicadas a seguir e executa as funções disponíveis, porém sem fazer *chain-of-thought*, sendo uma opção mais direta e barata.

Com o objetivo de formatar as saídas do modelo e orientar quanto ao que deve ser feito, foram implementadas as melhores estratégias de prompt 2.5.3, sendo elas: **(S1) Role Prompting** 1, **(S3) Abstração Não Técnica** 2, **(S6) Few-Shot Learning** 3 e **(S7) Negative Prompting** 4. O prompt completo pode ser encontrado no apêndice B.

Apesar da estratégia **S1** 1 não ter gerado resultados relevantes no estudo de referência (KOURANI et al., 2024), ela foi implementada uma vez que, diferente do estudo, não estamos tentando fazer análises em modelos BPMN e, sim, análises em cima de um contexto de negócio, então, sem o devido direcionamento, o LLM pode não entender o propósito.

Quadro 1 – (S1) Role Prompting

```
You are an expert Process Mining assistant. Your primary goal is to help users
analyze and understand event logs by using the tools at your disposal.
```

A estratégia **S3** 2 faz com que seja evitada a utilização do jargão técnico da área, aumentando a explicabilidade do sistema. Com esta, foi adicionada uma instrução para traduzir a mensagem final na linguagem do usuário, uma vez que a tendência é o LLM responder em inglês, prejudicando o entendimento da explicação final. Por fim, foi adicionada uma instrução para realizar um resumo e sugestão ao fim da resposta, forçando a tornar a resposta mais acionável para o usuário e extrair insights.

Quadro 2 – (S3) Abstração Não Técnica, abreviado com "..."para deixar compacto

```
Follow these rules strictly:
1. Translate the constraints rules and the activities to the language of the user.
2. Answer in natural language, so that any user not familiar with the process mining
 can understand your answer without any technical knowledge.
...
```

Ao fim, as estratégias **S6** 3 e **S7** 4 foram implementadas para guiar o LLM em como deve ser respondido, mostrando pelo menos 3 exemplos de resposta que podem ser vistos no B. Uma última instrução adicionada foi a de responder com base no contexto informado previamente, no qual esse contexto é acoplado no próprio modelo antes de receber os prompts e, neste trabalho, ele se refere à própria descrição do dataset.

Quadro 3 – (S6) Few-Shot Learning, abreviado com "..."para deixar compacto

```
Here are some examples of how you should respond based on the conversation history:

**EXAMPLE 1: ANOMALIES FOUND**

*Question:*

Quais são as anomalias para a regra de início e coexistência?

*Tool Responses:*
{'Start[register request]': 5}
{'Co-Existence[approve order, ship goods]': 12}

*Correct Final Answer:*
Olá! Analisei o log de eventos e encontrei as seguintes violações de regras:

* A regra de *Início* foi encontrada *5* violações para a atividade ...
* A regra de *Coexistência* foi encontrada *12* violações para as ...

Isso sugere que alguns casos não começam com o registro da solicitação ...

---
**EXAMPLE 2: ANOMALY FOUND** ...
**EXAMPLE 3: NO ANOMALIES FOUND** ...
```

Quadro 4 – (S7) Negative Prompting, abreviado com '...' para deixar compacto

```
Here are some examples of how you should NOT respond:

**EXAMPLE 1: NOT TRANSLATED**

*Question:*
Quais são as anomalias para a regra de início e coexistência?

*Tool Responses:*
{'Start[register request]': 5}
{'Co-Existence[approve order, ship goods]': 12}

*Incorrect Final Answer:*
Hello! I analyzed the event log and found the following rule violations:
* The *Start* rule had *5* violations for the activity *register request*.
* The *Co-Existence* rule had *12* violations for the activities ...

** EXAMPLE 2: TECHNICAL JARGON** ...
```

O algoritmo de análise de conformidade escolhido para essa arquitetura é o algoritmo de detecção de anomalias xSemAD 2.4, pois, além de não precisar de um modelo de referência (facilitando seu uso), exibe quais restrições foram violadas, fornecendo uma base para a LLM inferir insights e ações práticas a partir das não conformidades.

Para conseguir executar essa técnica através do LLM e cumprir o objetivo **OE2** 1.2.1, é criada uma função [3] que serve como ponte. Essa função é fornecida como *tool* para o LLM, no qual possui uma descrição com a sua finalidade, as especificações dos argumentos e o retorno da mesma para o LLM aprender sobre ela e executá-la corretamente.

[3] Código fonte disponível em: <https://github.com/JDaniloC/Projeto-2025-IF691>

Já o modelo agente *ReAct* recebe um ambiente de execução *Python* como *ferramenta*, não conseguindo ver as funções diretamente; por isso, foi adicionada uma instrução para observar quais funções estão disponíveis em memória e aprender sobre elas por meio do comando que exibe as descrições, como se pode ver no quadro 5.

Quadro 5 – ReAct Prompt, abreviado com "..."para deixar compacto

```
[ROLE PROMPT] ...
[ReAct INSTRUCTIONS] ...

## Some rules to follow:
1. **Use Available Functions:** Do dir() to see the available functions and help()
to learn about them. Use them to answer the user's question.
...
```

Por fim, é importante ressaltar que, mesmo antes de inicializar o modelo de linguagem, o dataset 3.2 é carregado em memória, ficando disponível para a LLM utilizar. No fim, essa é a sequência de eventos que ocorre ao iniciar o sistema:

1. O log de eventos é carregado em memória em conjunto com o modelo *events2constraints* que é necessário para execução da técnica xSemAD.

2. O modelo de linguagem é inicializado com as instruções iniciais.

3. O usuário envia uma pergunta que é analisada pelo modelo de linguagem, no sentido de observar se é uma pergunta sobre detecção de anomalias. Se for o caso, e essa detecção poder ser executada:

   - **Modelo de resposta direta:** Chama o algoritmo xSemAD com os argumentos inferidos a partir da pergunta do usuário.
   - **Caso seja o modelo de agente:**

     a) Analisa quais são as funções disponíveis e avalia cada uma delas.

     b) Executa o algoritmo xSemAD com os argumentos inferidos a partir da pergunta do usuário.

     c) Reflete em relação a resposta com base no contexto fornecido.

     d) Repete o passo anterior até que a resposta esteja cumprindo o que foi pedido.

4. É devolvida uma resposta explicando o resultado, no formato pedido.

4 RESULTADOS

O artefato permite receber uma pergunta em linguagem natural, como por exemplo "Diga quais as anomalias que se referem ao início do processo". Essa pergunta é interpretada pelo modelo e transformada em uma forma que possa executar o algoritmo de detecção de anomalias com os argumentos corretos, como pode ser visto no quadro 6:

Quadro 6 – Reposta do algoritmo para 'Diga quais as anomalias que se referem ao início do processo.'

```
>>> analisar_anomalias_de_processo(constraint_type='Init', threshold=0.75)
Get prediction for constraint type:  Init
>>> {'Init[send␣reminder]': 6449, 'Init[declaration␣approved␣by␣supervisor]': 6449,
'Init[declaration␣rejected␣by␣supervisor]': 6449, 'Init[request␣payment]': 6449}
```

Perceba que o gatilho "início do processo" foi interpretado corretamente como uma restrição de "Init" do DECLARE, então foi executado o algoritmo de detecção de anomalias com os parâmetros padrão (limiar igual a 75%), que devolve 4 (quatro) anomalias como resultado. No caso do modelo de resposta direta, o modelo simplesmente avalia a resposta do algoritmo e já a formata conforme o que foi pedido, resultando no quadro 7:

Quadro 7 – Resposta do prompt 'Diga quais as anomalias que se referem ao início do processo.'

```
Olá! Analisei o log de eventos e encontrei as seguintes violações para a regra de **
Início**:

*   A atividade **enviar lembrete** iniciou o processo indevidamente **6449** vezes.
*   A atividade **declaração aprovada pelo supervisor** iniciou o processo
indevidamente **6449** vezes.
*   A atividade **declaração rejeitada pelo supervisor** iniciou o processo
indevidamente **6449** vezes.
*   A atividade **solicitar pagamento** iniciou o processo indevidamente **6449**
vezes.

Isso indica que um número significativo de solicitações de despesas de viagem não
está começando com a submissão de um pedido, como seria o esperado. Em vez disso, os
 processos estão começando com etapas intermediárias ou finais, como o envio de
lembretes ou a aprovação de uma declaração, o que sugere que o fluxo do processo não
 está sendo seguido corretamente desde o princípio.
```

Perceba que, assim como pedido, o modelo devolveu uma resposta na linguagem do usuário (português), listando cada uma das restrições sem utilizar o jargão técnico, mas traduzindo para uma forma aplicada ao problema, trocando "Init(NOME)" por "A atividade NOME iniciou".

Por fim, faz uma análise que caracteriza cada uma das restrições encontradas para o que foi enviado como contexto, no qual cita "não está começando com a **submissão de um pedido**, como seria o esperado", que é uma informação de uma atividade presente apenas no contexto e, então, listando em seguida as restrições que quebram esse contexto.

Ao fazer a mesma pergunta para o modelo de agente ReAct, existe uma diferença substancial, onde o modelo realiza as etapas "Thought" - "Action" - "Action Input" repetidas vezes até chegar ao resultado final. Essas etapas basicamente se referem à transcrição do pensamento do modelo, decisão quanto à ação que deve ser tomada e à ação, respectivamente.

Quadro 8 – Etapas do ReAct (chain-of-thoughts) do modelo

```
Thought: O usuário está perguntando sobre anomalias no início do processo. Para
responder a essa pergunta, preciso verificar as violações da regra de "Início". Vou
usar as ferramentas disponíveis para encontrar essas violações e, em seguida,
apresentar a resposta de forma clara e em português. Primeiro, vou listar as funções
 disponíveis para ver qual delas pode me ajudar.
Action: Stateful Python Interpreter
Action Input: dir()

=> ['__builtins__', 'analyse_eventlog_anomalies', 'get_eventlog']

Thought: O usuário quer saber sobre anomalias no início do processo. A função `
analyse_eventlog_anomalies` parece ser a mais adequada para isso. Vou verificar a
documentação dessa função para entender como usá-la para encontrar violações da
regra de "Início".
Action: Stateful Python Interpreter
Action Input: help(analyse_eventlog_anomalies)

[RESULTADO DA FUNÇÃO HELP]

Thought: O usuário quer saber sobre as anomalias no início do processo. A documentaç
ão da função `analyse_eventlog_anomalies` indica que o `constraint_type` 'Init' é
usado para verificar as regras de início. Primeiro, preciso carregar o log de
eventos usando a função `get_eventlog()`. Depois, chamarei a função `
analyse_eventlog_anomalies` com o log carregado e o `constraint_type` definido como
'Init'. Finalmente, traduzirei o resultado para uma resposta clara e em português.
Action: Stateful Python Interpreter
Action Input: analyse_eventlog_anomalies(get_eventlog(), constraint_type='Init')

[RESULTADO DO ALGORITMO]
```

Perceba que, diferente do modelo de resposta direta, este modelo precisa listar quais funções estão disponíveis, pois a única *ferramenta* que tem acesso é o Python Interpreter. O resultado é exibido no quadro 9, que se demonstra mais sucinto do que o modelo anterior, porém mantendo o mesmo valor de negócio e chega ao mesmo resultado.

Quadro 9 – Resposta do prompt 'Diga quais as anomalias que se referem ao início do processo.' do agente

```
Olá! Analisei o log de eventos para encontrar anomalias relacionadas ao início do
processo e encontrei as seguintes violações:

A regra de *Início* foi violada 6.449 vezes para cada uma das seguintes atividades:
*   *enviar lembrete*
*   *declaração aprovada pelo supervisor*
*   *declaração rejeitada pelo supervisor*
*   *solicitar pagamento*

Isso indica que um número significativo de casos não está começando com a atividade
esperada, que seria a submissão de uma solicitação pelo funcionário. Em vez disso,
eles estão começando com atividades que deveriam ocorrer mais tarde no processo,
como aprovações, rejeições ou lembretes.
```

Os modelos também conseguem receber parâmetros numéricos, que é o caso do limiar de aceitação, como mostrado no quadro 10. Perceba que ele executou a função com o *threshold* como **0.85** que representa os 85% que foram pedidos. Além disso, é perceptível o impacto positivo da explicação quando o resultado é vazio, uma vez que, para um usuário não técnico, um resultado vazio pode ser difícil de entender.

Quadro 10 – Resposta do prompt 'Detecte as anomalias de fim de processo com um limiar de 85%'

```
>>> analyse_eventlog_anomalies(constraint_type='End', threshold=0.85)
>>> {}

Ótimas notícias! Após analisar o log de eventos, não encontrei nenhuma violação para
 as regras do tipo *Fim* com o limiar de 85%.

Isso indica que o processo está terminando consistentemente com as atividades
esperadas.
```

Além disso, é possível executar mais de uma operação por vez, como no exemplo do quadro 11, que obtém o resultado de duas análises e filtra a resposta conforme o que foi pedido:

Quadro 11 – Resposta do prompt 'Detecte as anomalias as maiores anomalias do tipo Succession e Alternate Precedence.' pelo modelo do tipo agente

```
Olá! Analisei o log de eventos para encontrar as anomalias mais significativas para
as regras de **Sucessão** e **Precedência Alternada**. Abaixo estão os resultados:

### Regra de Sucessão
A regra de **Sucessão** verifica se a atividade A ocorre se, e somente se, for
seguida (ou precedida) pela atividade B no processo. A maior anomalia encontrada foi
:

*   **Sucessão (solicitar pagamento, declaração aprovada pelo supervisor)**: **6.183
 violações**

Isso indica que as atividades *solicitar pagamento* e *declaração aprovada pelo
supervisor* raramente ocorrem juntas e em sequência. O número muito alto de violaçõ
es sugere que o fluxo normal do processo não exige que uma dessas atividades seja
imediatamente seguida pela outra, o que é esperado, já que pode haver outras etapas
entre elas ou elas podem não estar relacionadas no mesmo fluxo imediato.

### Regra de Precedência Alternada
A regra de **Precedência Alternada** verifica se uma atividade B é sempre precedida
por uma atividade A, não necessariamente de forma direta. As maiores anomalias
encontradas foram:

*   **Precedência Alternada (declaração aprovada pelo supervisor, permissão aprovada
 pelo supervisor)**: **637 violações**
*   **Precedência Alternada (solicitar pagamento, permissão aprovada pelo supervisor
)**: **637 violações**

Essas violações indicam que, em 637 casos, a atividade *permissão aprovada pelo
supervisor* ocorreu sem ter sido precedida pelas atividades *declaração aprovada
pelo supervisor* ou *solicitar pagamento*. De acordo com o contexto do processo, as
permissões são necessárias para viagens internacionais e devem ser aprovadas *antes*
 da viagem e do pedido de reembolso. Portanto, este comportamento é esperado para
viagens internacionais e não representa necessariamente um erro no processo.
```

Observe que a pergunta é vaga, pois não especifica o que caracteriza uma "maior" anomalia. Esse critério também não foi definido no contexto apresentado ao modelo. Ainda assim, o modelo inferiu que "as anomalias mais significativas" seriam aquelas que possuem um "número muito alto de violações". Entretanto, vale ressaltar que o conceito de "maior" anomalia pode variar a depender do contexto.

Por exemplo, uma situação em que ocorre o reembolso dos custos de uma viagem sem aprovação prévia pode ser considerada mais grave, mesmo que ocorra com menor frequência. Portanto, os critérios para definir a relevância das anomalias devem ser estabelecidos explicitamente na camada de contexto fornecida para o modelo.

Ainda reforçando a importância do contexto na interpretação dos resultados, perceba que, em relação à regra de precedência alternada, o modelo corrigiu uma falsa acusação de anomalia feita pelo algoritmo. O algoritmo havia sinalizado como anômalas situações em que a "declaração" foi aprovada antes da "permissão" do supervisor, e em que "solicitar pagamento" foi realizado antes dessa mesma "permissão" do supervisor. Nestes casos, o modelo analisou o contexto fornecido e o referenciou para demonstrar que a ordem adotada — em que é necessário obter a permissão do supervisor antes de qualquer ação, como a solicitação de pagamento — está, na verdade, em conformidade com a regra estabelecida.

Quadro 12 – Resposta do prompt 'Verifique se existe alguma anomalia de Succession para as atividades 'declaration approved by supervisor' e 'request payment'

```
Olá! Analisei o log de eventos para as anomalias de *Sucessão* e encontrei algumas
violações, incluindo as que você solicitou.

A regra de *Sucessão* significa que uma atividade deve ocorrer se, e somente se, for
 imediatamente seguida ou precedida por outra atividade específica.

Para as atividades que você mencionou, encontrei os seguintes problemas:
*   **Declaração aprovada pelo supervisor** e **solicitar pagamento**: Foram
encontradas **244** violações. Isso significa que em 244 casos, a aprovação pelo
supervisor não foi seguida imediatamente pela solicitação de pagamento, ou vice-
versa.
*   **Solicitar pagamento** e **declaração aprovada pelo supervisor**: Foram
encontradas **6.183** violações. Este número elevado indica que a atividade de
solicitar pagamento frequentemente não é precedida pela aprovação do supervisor, ou
a aprovação não ocorre logo após a solicitação.

Além disso, a análise revelou outras violações de sucessão que podem ser do seu
interesse:
*   **Enviar lembrete** e **declaração aprovada pelo supervisor**: **406** violações
.
*   **Declaração rejeitada pelo supervisor** e **declaração aprovada pelo supervisor
**: **122** violações.

Esses números, especialmente o de 6.183, sugerem que o processo real se desvia
bastante da regra de que a solicitação de pagamento e a aprovação do supervisor
devem ocorrer em sequência imediata. Pode ser útil investigar por que essas
atividades não estão ocorrendo juntas como esperado.
```

O quadro 12 relata o teste que realiza uma busca de anomalia a nível de evento, nesse caso, foi pedido para buscar a anomalia da restrição *Succession* para as atividades *declaration approved by supervisor* e *request payment*, em que não apenas foi apontado o resultado desejado pela busca, mas também não ficou preso à ordem das atividades, mostrando tanto o caso onde a "declaration approved by supervisor" veio primeiro que "request payment" como o contrário; observe que, ainda que o nome da restrição e das atividades estivessem em inglês, a resposta foi devolvida em português, realizando a filtragem que foi pedida e, por fim, exibindo e avaliando as demais anomalias não cobertas pela filtragem.

Tabela 3 – Selected anomalies detected for the international declarations log

id	detected anomalies by xSemAD	Freq
a1	Alternate Response(declaration approved by supervisor, end trip)	9
a2	Response(declaration rejected by supervisor, declaration approved by supervisor)	114
a3	Succession(declaration approved by supervisor, request payment)	244
a4	Alternate Precedence(request payment, declaration approved by supervisor)	246

Fonte: BUSCH; KAMPIK; LEOPOLD (2024)

Por fim, ao analisar esse mesmo dataset para demonstrar o potencial real do algoritmo xSemAD, os pesquisadores Busch et al. (BUSCH; KAMPIK; LEOPOLD, 2024) chegaram aos resultados mostrados na tabela 3 e, a partir desses resultados, extraíram os seguintes insights:

> "*a1* destaca a necessidade de aprovações antes do fim da viagem. *a2* mostra um mecanismo corretivo onde declarações rejeitadas são eventualmente aprovadas após ajustes. *a3* e *a4* confirmam a necessidade de aprovações antes das solicitações de pagamento."
> *tradução livre de* (BUSCH; KAMPIK; LEOPOLD, 2024)

Quadro 13 – Resposta do prompt 'Detecte as anomalias do tipo Response.'

```
Olá! Analisei o log de eventos para encontrar violações da regra de "Resposta" e
encontrei o seguinte:

- A regra de **Resposta** foi violada **114 vezes** para a sequência de atividades
*"declaração rejeitada pelo supervisor"* e *"declaração aprovada pelo supervisor"*.

A regra de "Resposta" significa que, se uma atividade ocorre, outra atividade especí
fica deve ocorrer em seguida. Neste caso, a análise mostra que em 114 situações em
que uma declaração foi rejeitada pelo supervisor, não houve uma aprovação
subsequente como o modelo esperava.

No entanto, é importante notar que, no contexto deste processo, uma rejeição não
deveria ser seguida por uma aprovação. O alto número de violações aqui provavelmente
 indica uma peculiaridade no fluxo do processo ou um problema na forma como os dados
 são registrados, em vez de um erro real dos funcionários.
```

A partir desses insights, podemos comparar as impressões humanas com as impressões geradas pelo modelo. Executando arbitrariamente o algoritmo para a regra *Response* (a2) podemos ver o resultado mostrado no quadro 13. O primeiro ponto observado é que, assim como nos exemplos anteriores, o modelo decidiu explicar a regra da restrição primeiro para depois realizar a análise, algo que é muito importante para os usuários não técnicos.

Além disso, diferente da análise humana que considerou a quebra de restrição como uma ação corretiva de problemas comuns ("É mostrado um mecanismo corretivo onde declarações rejeitadas são eventualmente aprovadas após ajustes."), o modelo considerou que pode ser um problema de fluxo ou nos dados uma vez que considera o número de ocorrências arbitrariamente alto, fornecendo uma outra visão da análise.

Enfim, os modelos demonstraram vasta utilidade e versatilidade para responder às mais diversas perguntas, destacando-se, entre outros aspectos, pelos seguintes fatores:

- Permite executar algoritmos de análise de conformidade, neste caso o xSemAD, recebendo as perguntas em linguagem natural e sendo respondido da mesma forma.
- Traduz o jargão técnico, explica e infere insights a partir dos resultados do algoritmo no idioma do usuário, conforme o que foi pedido nas instruções iniciais, seguindo as técnicas de prompting 2.5.3 e as instruções extras B.
- Interpreta as restrições mesmo quando descritas apenas em linguagem natural, e quando necessário, explica a lógica da regra para o usuário.
- Identifica quando o usuário quer modificar algum outro parâmetro do algoritmo e o executa passando os argumentos com a tipagem correta.
- Realiza operações mais robustas e refinadas como pedir para analisar mais de uma restrição, filtrar pelas anomalias mais frequentes ou realizar buscas a nível de evento.
- Avaliar e corrigir se necessário os resultados do algoritmo conforme o contexto fornecido, como foi visto no exemplo do quadro 11.
- Devolver conclusões valiosas refinando os resultados do algoritmo com base no contexto.

Ou seja, essa abordagem resolve os problemas elencados ao facilitar o uso de técnicas de análise de conformidade e fornecer resultados que podem ser entendidos por pessoas que não são especialistas em mineração de processos, democratizando sua utilização e possibilitando a extração de ações práticas a partir dessas técnicas.

5 DISCUSSÃO E TRABALHOS FUTUROS

5.1 DISCUSSÃO

Um exemplo real que poderia se beneficiar deste framework pode ser visto na tentativa de realizar a análise de conformidade pelo estudo (ALEXANDER et al., 2020), que analisou o mesmo *dataset* 3.2 utilizado neste projeto. Esse exemplo é pertinente, pois esbarra em todos os problemas que foram citados neste trabalho, são eles:

1. **Necessidade de criação de um modelo de referência:** Eles precisaram criar um modelo de referência em BPMN com base na documentação do dataset.
2. **Dificuldade de uso da análise de conformidade:** Eles precisaram converter o modelo BPMN em rede de Petri para executar técnicas de análise de conformidade (TBR e Alignments 2.1), avaliando a conformidade dos processos com o modelo de referência.
3. **Resultado insuficiente e não acionável:** O resultado em ambos os algoritmos para "fitness" geral foi muito baixo, em torno de 10%, para todo o log de eventos. Para tentar explicar o baixo desempenho, foi apontado que as especificações fornecidas pela documentação eram imprecisas e não mencionavam todas as atividades.

Perceba que especialistas da área de mineração de processos tiveram que fazer o esforço de analisar um log de eventos real, criar um modelo de referência a partir de uma descrição vaga do log de eventos, converter esse modelo, executar técnicas tradicionais de análise de conformidade e, mesmo assim, não receberam um resultado satisfatório. Por outro lado, se esse framework fosse utilizado nesse estudo, eles teriam uma análise de conformidade mais simples de utilizar, mais explicável e acionável.

A maioria dos avanços da área que é interseção entre mineração de processos e LLMs vão na direção de utilizar LLMs para calcular métricas de mineração de processos, substituir abordagens tradicionais ou melhorar os algoritmos existentes. Apesar de muito úteis, essas abordagens deixam de lado um grande potencial fornecido pelas LLMs no sentido de facilitar o entendimento dos resultados dos algoritmos tradicionais, como provado neste trabalho.

5.2 TRABALHOS FUTUROS

Para tornar esse estudo cientificamente mais robusto, é possível realizar uma estratégia de autoavaliação da LLM (BERTI et al., 2024), na qual o resultado de uma sessão é fornecido a outra LLM ou a uma sessão diferente da mesma LLM para pontuar a qualidade da saída. A pontuação seria definida por critérios como clareza de raciocínio, ausência de alucinações, explicação aderente ao contexto, entre outros, utilizando os scripts de autoavaliação do trabalho de Kourani et al. (KOURANI et al., 2024) para possibilitar gerar muitas respostas.

Para provar que a abordagem é útil para pessoas não técnicas, faz-se necessário realizar um estudo de caso que aplique a abordagem em um contexto real. Neste caso, pessoas de uma área de negócio sem conhecimento ou com pouco conhecimento de mineração de processos, como o judiciário, realizariam tarefas de mineração de processos utilizando a abordagem tradicional - por meio de ferramentas já existentes - se comparado com a interface em linguagem natural, onde, ao final, os participantes avaliam as duas abordagens em termos de facilidade de uso, potencial para descoberta de ações práticas, entre outros critérios.

Uma vez que a abordagem apresentada nesse trabalho possibilita expandir para adicionar outras técnicas de mineração de processos, então seria interessante possibilitar o uso das técnicas presentes na biblioteca de ferramentas Process Mining for Python (PM4Py) (BERTI; ZELST; AALST, 2019) possibilitando análises como obter informações do log de eventos e até mesmo realizar a descoberta de modelos de processos.

Por fim, além deste conjunto de ferramentas, é também interessante aplicar no framework as técnicas presentes na biblioteca *Declare4Py* (DONADELLO et al., 2022) expandindo a análise para modelos DECLARE que se mostraram interpretáveis por esse estudo, uma vez que a técnica analisada 2.4 utiliza restrições desse tipo.

6 CONCLUSÃO

Neste trabalho, foram apresentados alguns problemas enfrentados ao tentar realizar uma análise de conformidade: necessidade da criação de um modelo de referência, conhecimento técnico para conseguir executar os algoritmos, resultados insuficientes para se traduzir em ações práticas ou difíceis de interpretar. Esses problemas somados inviabilizam o uso dessas técnicas por pessoas não técnicas na área de mineração de processos.

Para solucionar esses problemas, foi utilizada a técnica xSemAD por meio de uma interface em linguagem natural, provando que é possível fornecer uma forma pela qual pessoas que não são da área de mineração de processos podem realizar a análise de conformidade, e respondendo à pergunta central da pesquisa "Como a utilização de modelos de LLMs pode enriquecer e facilitar o entendimento da análise de conformidade?" da seguinte forma:

É possível utilizar as LLMs como meio para facilitar o uso das técnicas já existentes, traduzir o jargão técnico da área de forma a ser entendido por usuários leigos e transformar os resultados desses algoritmos em ações práticas, democratizando o uso dessas técnicas e tornando a análise de conformidade explicável e acionável para usuários não técnicos.

Para isso, foi fornecida uma arquitetura e abordagem capazes de receber das mais diversas técnicas de mineração de processos, utilizando a técnica xSemAD como exemplo e comparando os resultados com a abordagem tradicional. Em resumo, por meio dessa abordagem foi possibilitado expandir os limites da análise de conformidade e apresentado uma forma de como aplicar as técnicas de mineração de processos para o público em geral.

REFERÊNCIAS

AALST, W. van der. Process mining: Overview and opportunities. *ACM Trans. Manage. Inf. Syst.*, Association for Computing Machinery, New York, NY, USA, v. 3, n. 2, jul. 2012. ISSN 2158-656X. Disponível em: <https://doi.org/10.1145/2229156.2229157>.

ALEXANDER, N.; EVGENIYA, S.; YULIA, K.; ANNA, S.; YURI, T. Bpi challenge 2020 report: Analyzing international and domestic travel processes. In: *Process Mining Conference 2020, Central Chernozem Bank PJSC Sberbank January*. [S.l.: s.n.], 2020.

BARBIERI, L.; MADEIRA, E.; STROEH, K.; AALST, W. van der. A natural language querying interface for process mining. *Journal of Intelligent Information Systems*, v. 61, n. 1, p. 113–142, aug 2023. ISSN 1573-7675. Disponível em: <https://doi.org/10.1007/s10844-022-00759-9>.

BARBIERI, L.; STROEH, K.; MADEIRA, E. R. M.; AALST, W. M. P. van der. An llm-based q&a natural language interface to process mining. In: DELGADO, A.; SLAATS, T. (Ed.). *Process Mining Workshops*. Cham: Springer Nature Switzerland, 2025. p. 5–17. ISBN 978-3-031-82225-4.

BERTI, A.; KOURANI, H.; HäFKE, H.; LI, C.-Y.; SCHUSTER, D. Evaluating large language models in process mining: Capabilities, benchmarks, and evaluation strategies. In: _____. *Enterprise, Business-Process and Information Systems Modeling*. Springer Nature Switzerland, 2024. p. 13–21. ISBN 9783031610073. Disponível em: <http://dx.doi.org/10.1007/978-3-031-61007-3_2>.

BERTI, A.; SCHUSTER, D.; AALST, W. M. P. van der. Abstractions, scenarios, and prompt definitions for process mining with llms: A case study. In: WEERDT, J. D.; PUFAHL, L. (Ed.). *Business Process Management Workshops*. Cham: Springer Nature Switzerland, 2024. p. 427–439. ISBN 978-3-031-50974-2.

BERTI, A.; ZELST, S. J. van; AALST, W. van der. *Process Mining for Python (PM4Py): Bridging the Gap Between Process- and Data Science*. 2019. Disponível em: <https://arxiv.org/abs/1905.06169>.

BEZERRA, F.; WAINER, J.; AALST, W. M. P. van der. Anomaly detection using process mining. In: HALPIN, T.; KROGSTIE, J.; NURCAN, S.; PROPER, E.; SCHMIDT, R.; SOFFER, P.; UKOR, R. (Ed.). *Enterprise, Business-Process and Information Systems Modeling*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009. p. 149–161. ISBN 978-3-642-01862-6. Disponível em: <https://www.vdaalst.com/publications/p512.pdf>.

BUSCH, K.; KAMPIK, T.; LEOPOLD, H. *xSemAD: Explainable Semantic Anomaly Detection in Event Logs Using Sequence-to-Sequence Models*. 2024. Disponível em: <https://arxiv.org/abs/2406.19763>.

CASPARY, J.; REBMANN, A.; AA, H. van der. Does this make sense? machine learning-based detection of semantic anomalies in business processes. In: FRANCESCOMARINO, C. D.; BURATTIN, A.; JANIESCH, C.; SADIQ, S. (Ed.). *Business Process Management*. Cham: Springer Nature Switzerland, 2023. p. 163–179. ISBN 978-3-031-41620-0.

CICCIO, C. D.; MONTALI, M. Declarative process specifications: Reasoning, discovery, monitoring. In: _____. *Process Mining Handbook*. Cham: Springer International

Publishing, 2022. p. 108–152. ISBN 978-3-031-08848-3. Disponível em: <https://doi.org/10.1007/978-3-031-08848-3_4>.

DONADELLO, I.; RIVA, F.; MAGGI, F. M.; SHIKHIZADA, A. Declare4py: A python library for declarative process mining. In: *BPM (PhD/Demos)*. [S.l.]: CEUR-WS.org, 2022. (CEUR Workshop Proceedings, v. 3216), p. 117–121.

DUNZER, S.; STIERLE, M.; MATZNER, M.; BAIER, S. Conformance checking: a state-of-the-art literature review. In: *Proceedings of the 11th International Conference on Subject-Oriented Business Process Management*. New York, NY, USA: Association for Computing Machinery, 2019. (S-BPM ONE '19). ISBN 9781450362504. Disponível em: <https://doi.org/10.1145/3329007.3329014>.

HUANG, X.; LIU, W.; CHEN, X.; WANG, X.; WANG, H.; LIAN, D.; WANG, Y.; TANG, R.; CHEN, E. *Understanding the planning of LLM agents: A survey*. 2024. Disponível em: <https://arxiv.org/abs/2402.02716>.

KLEIN, S.; LAHANN, J.; MAYER, L.; NEU, D.; PFEIFFER, P.; REBMANN, A.; SCHEID, M.; WILLEMS, B.; FETTKE, P. Business process intelligence challenge 2020: Analysis and evaluation of a travel process. In: *10th business process intelligence challenge at the int. conf. on process mining*. [S.l.: s.n.], 2020.

KOURANI, H.; BERTI, A.; HENNRICH, J.; KRATSCH, W.; WEIDLICH, R.; LI, C.-Y.; ARSLAN, A.; SCHUSTER, D.; AALST, W. M. P. van der. *Leveraging Large Language Models for Enhanced Process Model Comprehension*. 2024. Disponível em: <https://arxiv.org/abs/2408.08892>.

MINAEE, S.; MIKOLOV, T.; NIKZAD, N.; CHENAGHLU, M.; SOCHER, R.; AMATRIAIN, X.; GAO, J. *Large Language Models: A Survey*. 2025. Disponível em: <https://arxiv.org/abs/2402.06196>.

REBMANN, A.; SCHMIDT, F. D.; GLAVAš, G.; AA, H. van der. *On the Potential of Large Language Models to Solve Semantics-Aware Process Mining Tasks*. 2025. Disponível em: <https://arxiv.org/abs/2504.21074>.

van der Aa, H.; REBMANN, A.; LEOPOLD, H. Natural language-based detection of semantic execution anomalies in event logs. *Information Systems*, v. 102, p. 101824, 2021. ISSN 0306-4379. Disponível em: <https://www.sciencedirect.com/science/article/pii/S030643792100065X>.

YAO, S.; ZHAO, J.; YU, D.; DU, N.; SHAFRAN, I.; NARASIMHAN, K.; CAO, Y. React: Synergizing reasoning and acting in language models. In: *International Conference on Learning Representations (ICLR)*. [S.l.: s.n.], 2023.

ZICHE, C.; APRUZZESE, G. Llm4pm: A case study on using large language models for process modeling in enterprise organizations. In: _____. *Business Process Management: Blockchain, Robotic Process Automation, Central and Eastern European, Educators and Industry Forum*. Springer Nature Switzerland, 2024. p. 472–483. ISBN 9783031704451. Disponível em: <http://dx.doi.org/10.1007/978-3-031-70445-1_35>.

APÊNDICE A – DIRECT ANSWER PROMPT

```
You are an expert Process Mining assistant. Your primary goal is to help users
analyze and understand event logs by using the tools at your disposal.

Follow these rules strictly:
1. Translate the constraints rules and the activities to the language of the user.
2. Answer in natural language, so that any user not familiar with the process mining
 can understand your answer without any technical knowledge.
3. If context is provided, use it to enhance your response and judge the correctness
 of the tool responses.
4. End with a brief summary or suggestion, if applicable.

Here are some examples of how you should respond based on the conversation history:

**EXAMPLE 1: ANOMALIES FOUND**
*Question:*
Quais são as anomalias para a regra de início e coexistência?

*Tool Responses:*
{'Start[register request]': 5}
{'Co-Existence[approve order, ship goods]': 12}

*Correct Final Answer:*
Olá! Analisei o log de eventos e encontrei as seguintes violações de regras:

* A regra de *Início* foi encontrada *5* violações para a atividade *registrar
solicitação*.
* A regra de *Coexistência* foi encontrada *12* violações para as atividades *
aprovar pedido* e *enviar mercadoria*.

Isso sugere que alguns casos não começam com o registro da solicitação e que a
aprovação do pedido e envio da mercadoria nem sempre ocorrem juntas como deveriam.

---
**EXAMPLE 2: ANOMALY FOUND**
*Question:*
Quais são as violações para a regra de Precedência?

*Tool Response:*
{'Precedence[register request, approve order]': 3}

*Correct Final Answer:*
Olá! Após analisar o log de eventos, encontrei as seguintes violações para a regra
de *Precedência*:
```

```
* A regra de *Precedência* foi violada *3* vezes para a sequência de atividades *
registrar solicitação* e *aprovar pedido*.

Isso indica que, em alguns casos, está registrando uma solicitação sem que a
atividade de aprovação do pedido tenha ocorrido antes.

---
**EXAMPLE 3: NO ANOMALIES FOUND**
*Question:*
Verifique se existem violações para a regra de Sucessão.

*Tool Response:*
{}

*Correct Final Answer:*
Ótimas notícias! Após analisar o log de eventos, não encontrei nenhuma violação para
 as regras do tipo *Sucessão*.

O processo está em conformidade com todas as regras de sucessão esperadas.

Here are some examples of how you should NOT respond:

**EXAMPLE 1: NOT TRANSLATED**
*Question:*
Quais são as anomalias para a regra de início e coexistência?

*Tool Responses:*
{'Start[register request]': 5}
{'Co-Existence[approve order, ship goods]': 12}

*Incorrect Final Answer:*
Hello! I analyzed the event log and found the following rule violations:
* The *Start* rule had *5* violations for the activity *register request*.
* The *Co-Existence* rule had *12* violations for the activities *approve order* and
 *ship goods*.

** EXAMPLE 2: TECHNICAL JARGON**
*Question:*
Quais são as violações para a regra de Precedência?

*Tool Response:*
{'Precedence[register request, approve order]': 3}

*Incorrect Final Answer:*
Foi encontrada uma anomalia do tipo "Precedência" no par de eventos *registrar
solicitação* e *aprovar pedido*, ocorrendo *3* vezes.
```

APÊNDICE B – REACT PROMPT

```
[ROLE PROMPTING] ...

## Here are the context information about the event log you are analyzing:
{event_log_context}

## You have access to the following tools:
{tools}

## Use the following format:
Question: the input question you must answer
Thought: you should always think about what to do
Action: the action to take, should be one of [{tool_names}]
Action Input: the input to the action
Observation: the result of the action
... (this Thought/Action/Action Input/Observation can repeat N times)
Thought: I now know the final answer
Final Answer: the final answer to the original input question



## Some rules to follow:
1. **Use Available Functions:** Do dir() to see the available functions and help()
to learn about them. Use them to answer the user's question.
2.  **Translate:** Always translate the technical names of constraints and
activities (e.g., 'Init', 'Co-Existence', 'register request') into the user's
language (e.g., 'Início', 'Coexistência', 'registrar solicitação').
3. **Natural Language Responses:** Provide answers in natural language, ensuring
that any user, regardless of their familiarity with process mining, can understand
your response without technical jargon.
4.  **Analyze and Add Insight:** Do not just report the raw numbers from the tools.
Briefly analyze the eventlog and the activities involved to provide insights.
5. **Context Usage:** If context is provided, use it to enhance your response and
judge the correctness of the tool responses.

[EXAMPLES] ...

Begin!

Question: {input}
Thought: {agent_scratchpad}
```